1994-02

I. Circuitos digitales combinacionales

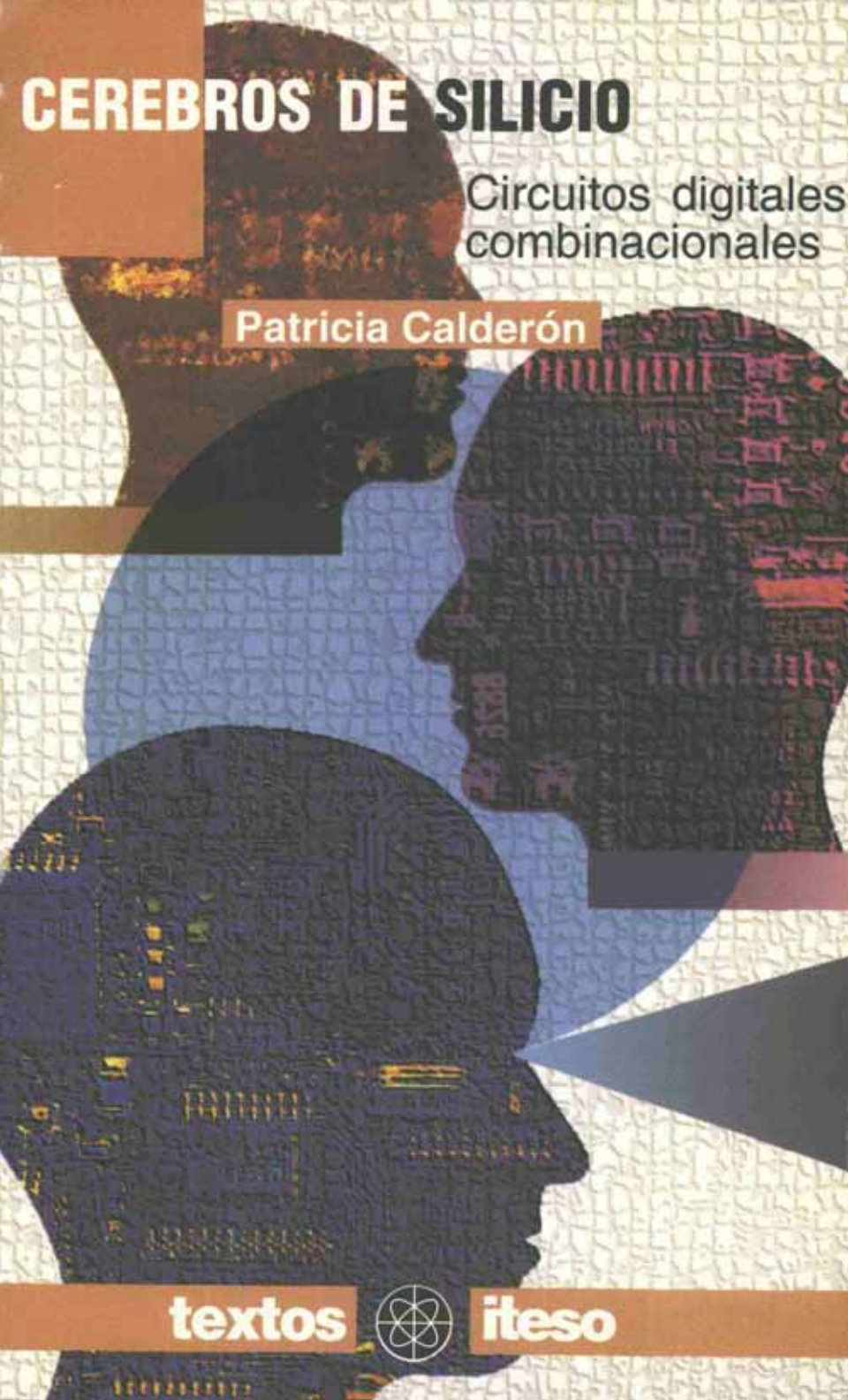
Calderón, Patricia

Calderón, P. (1994). Cerebros de silicio. I. Circuitos digitales combinacionales. Tlaquepaque, Jalisco: ITESO.

CEREBROS DE SILICIO

Circuitos digitales combinacionales

Patricia Calderón



textos iteso

Cerebros de silicio

Circuitos digitales combinacionales

Cerebros de silicio

Circuitos digitales combinacionales

Patricia Calderón

iteso

Portada: Margen \ JABAZ

ISBN 968-6101-33-0

Indice

Introducción

Este texto es la primera parte de un estudio teórico-práctico sobre los circuitos electrónicos digitales. Los temas expuestos a lo largo de este libro fueron elaborados para un curso dirigido a los alumnos de Ingeniería en Sistemas Computacionales del ITESO. A lo largo del curso se pretendió que, sin ningún prerrequisito, los alumnos finalizaran la experiencia académica sabiendo cómo se diseña un sistema digital y cómo funcionan los circuitos digitales, tanto en la teoría como en la práctica. El lector que tenga conocimientos básicos de electrónica o de lógica podrá dejar de leer algunos de los capítulos si así lo desea.

Para la utilización de este libro en cursos universitarios se recomienda un semestre para cubrir los capítulos I a VII. El segundo tomo de esta serie deberá tomar otro periodo académico igual.

Esta obra está compuesta por siete capítulos y un apéndice que se describirán brevemente a continuación. Si el lector está interesado en realizar las prácticas que se sugieren a lo largo del libro, es recomendable que primero se lea el apéndice que aparece al final. En él se muestra cómo diseñar una fuente, herramienta básica para realizar experimentos con circuitos digitales. También se incluye el diseño de un oscilador, para cuando se realicen experimentos con circuitos secuenciales.

En el capítulo I se presenta la ubicación de la materia: cuándo, dónde y para qué sirven los circuitos digitales. En el capítulo II se hace una relación de la utilidad y uso del álgebra de Boole en algunos ámbitos, incluyendo el de los circuitos digitales. También se muestra el primer método para minimizar y se hace énfasis en la importancia de saberlo hacer. El capítulo III es muy interesante porque muestra cómo trabajan los dispositivos de conmutación, es decir, cómo son las llamadas "cajas negras" por dentro.

El capítulo IV versa sobre sistemas numéricos. Se hace un recordatorio de los sistemas binario, octal y hexadecimal. Al final se invita a realizar una práctica utilizando un ALU (Unidad Lógico-aritmética). Los capítulos V y VI están dedicados a los métodos de minimización. Uno es gráfico y el otro es tabular. Se hace clara mención de cuándo es conveniente usar uno u otro.

Finalmente, en el capítulo VII se muestra cómo funcionan los circuitos combinacionales MSI, dónde se pueden utilizar y cómo diseñar con ellos sistemas más complejos.

Quiero darle las gracias a los alumnos que me ayudaron a revisar estos apuntes, a los encargados del laboratorio de *hardware* de Ingeniería en Sistemas Computacionales del ITESO por ayudarme a elaborar las prácticas y algunos de los dibujos, a los maestros que revisaron el material y al ITESO en general por las facilidades que ofrece a sus maestros para realizar este tipo de trabajos.

CAPÍTULO I
Circuitos digitales

Los circuitos digitales actualmente son dispositivos tan comunes que cualquiera puede manejarlos; por ejemplo, un niño jugando nintendo. Sin embargo no todos saben cómo funcionan. El objetivo de este capítulo es mostrar dónde se usan, sus ventajas y desventajas, cuándo un sistema es digital y cuándo no, cómo fueron evolucionando hasta llegar a ser lo que son ahora, cómo se clasifican y, de acuerdo con su clasificación, en qué parte de una computadora se encuentran.

Después de que se conozcan los circuitos digitales, dónde y cuándo se usan y lo importantes que son en los sistemas digitales, será más fácil e interesante estudiarlos.

1.1 Panorama general

Por lo regular se piensa que un circuito digital forma parte de una computadora, sin embargo su uso no está tan restringido, éste se encuentra en otros sistemas, simplemente un interruptor para apagar y prender la luz en nuestra casa es un circuito digital, mecánico por supuesto pero digital. Cualquier dispositivo que tenga dos estados, como prendido o apagado, activado o desac-

tivado, 1 ó 0, es un circuito digital binario, ya sea mecánico, eléctrico o electrónico.

Los dispositivos digitales electrónicos pueden formar parte de una computadora, así como también de un reloj digital, o un horno de microondas, etcétera. En este capítulo se define qué es un circuito digital y sus aplicaciones. Una de las aplicaciones más importante de estos circuitos está en las computadoras.

1.2 Clasificación de los sistemas

Los sistemas, por la forma en la que procesan las señales, se clasifican en analógicos y digitales. Una señal analógica es aquella transmisión o procesamiento de información en la cual la información varía entre 0 y x cantidad. Una señal digital es cualquier señal de transmisión o procesamiento de información, en la cual la información se representa por medio de cantidades físicas que se hallan tan restringidas que sólo pueden asumir valores discretos.

Si una señal está restringida a sólo dos valores discretos, entonces el sistema es binario. Un ejemplo de esto se puede observar en el mecanismo que se muestra en la figura 1.2.1.

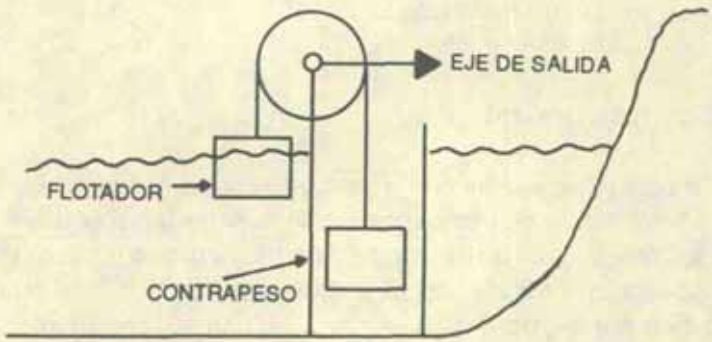


Figura 1.2.1. Mecanismo sensor del nivel del agua

En este sistema el eje de salida controla la posición de un potenciómetro. Este potenciómetro hace variar el voltaje de salida que se transmitirá por medio de dos alambres. En este caso el nivel del agua se mide con una variación de 0 a V. Este sistema es analógico. Sin embargo, tiene algunas desventajas. Por ejemplo la resistencia de un conductor varía con la temperatura, la humedad, incluso también con la longitud del alambre. El sistema es bueno cuando no se requiere mucha precisión, además es bastante económico. Ver figura 1.2.2.

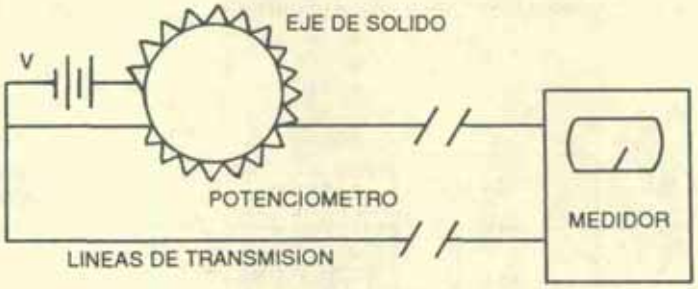


Figura 1.2.2. Sistema analógico para trasmitir el nivel del agua

Si se requiere de precisión el sistema de la figura 1.2.3 es el indicado. En este caso el eje de salida se conectará a un inte-

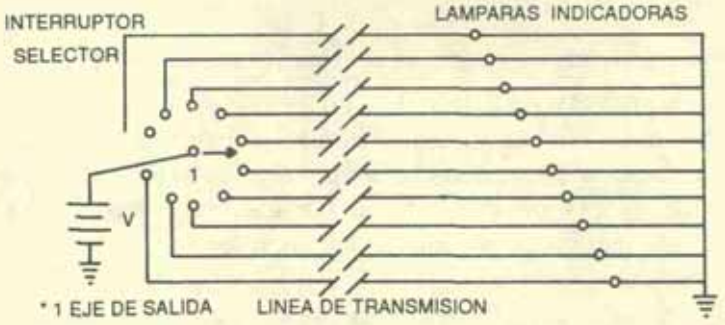


Figura 1.2.3. Sistema digital en paralelo para transmitir el nivel del agua

rruptor, a una fuente de voltaje y a una lámpara, la lámpara encendida indicará en qué nivel está el agua. En este caso, si la resistencia de los alambres aumenta debido a cualquier factor no habrá problema pues lo único que se requiere es que sea el suficiente para encender la lámpara, además no indicará mal el nivel del agua. Este es un sistema digital en paralelo. La desventaja de este sistema es que en vez de utilizar dos alambres se utilizan diez alambres y diez focos, lo que implica que sea muy costoso.

También se puede considerar el sistema digital de la figura 1.2.4. En este caso se tendrá un generador de pulsos, cuya función

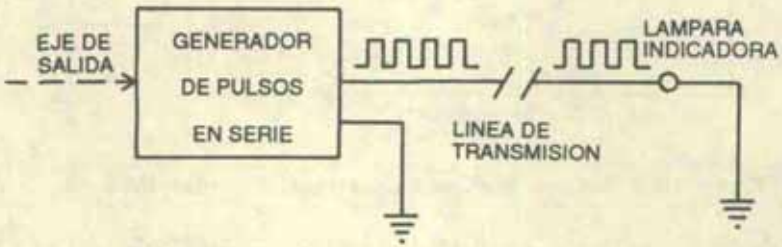


Figura 1.2.4. Sistema digital en serie para trasmitir el nivel del agua

es convertir el voltaje en pulsos en forma periódica. Los pulsos serán de uno a diez indicando el nivel del agua. Por supuesto, sería cuestión de que hubiera un operador que contara los pulsos en la lámpara indicadora. Puesto que los pulsos llegan uno a la vez, este sistema es un sistema digital en serie.

Si se comparan los dos sistemas digitales se podrá notar que las características principales del sistema en serie son que es

económico y lento. Las características principales del sistema en paralelo es que son más rápidos y costosos. Ver figura 1.2.5.

CARACTERÍSTICAS DE UN SISTEMA EN SERIE

- Económico
- Lento

CARACTERÍSTICAS DE UN SISTEMA EN PARALELO

- Más costoso
- Rápido

Figura 1.2.5. Comparación entre un sistema en serie y uno en paralelo

Un sistema intermedio se puede obtener utilizando un codificador. En éste la información de las 10 líneas se convierte a cuatro líneas. Cuando la información llega al operador ésta se puede convertir para que dé una lectura en un display. Ver figura 1.2.6.

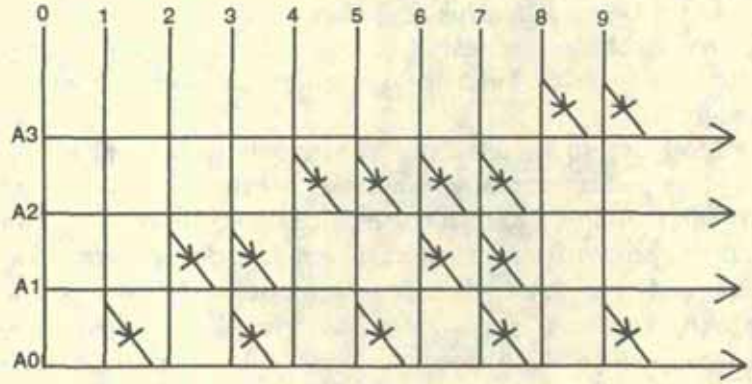


Figura 1.2.6. Codificador decimal a binario

1.3 Evolución de la electrónica digital

La electrónica digital es la base de las computadoras modernas, por eso vale la pena ver el desarrollo histórico de éstas, además de tener conocimiento de las personas cuya participación ha sido importante en los adelantos tecnológicos dentro de la computación.

A fines de 1940 el álgebra de lógica de George Boole, las tarjetas perforadas de Herman Hollerith y el calculador construido por George Aiken, fueron factores importantes en el avance de las computadoras.

Tiene también mucha importancia histórica Charles Babbage, aunque sus trabajos no influyeron directamente en el diseño de las computadoras modernas, las ideas básicas de este inventor del siglo XIX, tales como el almacenamiento de programa, aún subsisten. En 1812 Charles Babbage era profesor de matemáticas de la Universidad de Cambridge. Ahí desarrolló una máquina diferencial para efectuar automáticamente cálculos sencillos utilizados en las tablas de logaritmos y en las tablas trigonométricas. La máquina diferencial de Babbage ejecutaba una secuencia de operaciones, una a la vez; también concibió una máquina analítica que podía ejecutar una secuencia arbitraria de operaciones y tenía un almacenamiento interno de datos. La descripción que Babbage dio de esta máquina es muy similar al concepto actual sobre almacenamiento de programas de las computadoras modernas.

Las redes de interconexión por interruptores en una computadora pueden ser muy complejas, así como la lógica de un programa complicado puede ser difícil de analizar. El álgebra de Boole proporciona un método sistemático de representación y análisis. Boole fue un pionero en el campo de la lógica simbólica. Su libro *The laws of thought —Las leyes del pensamiento—* publicado en 1854 —¡hace más de cien años!— representa la lógica en forma de símbolos matemáticos y nos da las reglas para

detectar la verdad o la falsedad de una declaración. Para poder diseñar circuitos digitales debemos aprender álgebra de Boole.

La tarjeta perforada no es un componente de las computadoras electrónicas, sin embargo formó parte importante en el proceso de las computadoras y como tal se menciona en la historia. En el año de 1745 el francés Joseph M. Jaqcuard diseñó un método para utilizar los agujeros en unas tarjetas para controlar la selección de los hilos en los diseños de los tejidos en telares. Babbage también había propuesto agujeros en tarjetas para proporcionar datos de entrada a su máquina analítica.

Herman Hollerith fue el iniciador del uso de estas tarjetas perforadas en una máquina. La necesidad de un equipo mecánico fue visualizada por Hollerith al estar trabajando en el censo de 1890. Durante la década de los años de 1880 a 1890, desarrolló la idea de la tarjeta perforada y diseñó una máquina para el censo. Se seleccionó el método de Hollerith para tabular el censo de 1890; la tabulación fue rápida y hubo un ahorro considerable con respecto al método manual anterior. En el año de 1896 Hollerith organizó su Tabulating Machine Company para la manufactura y venta de máquinas y tarjetas. Esta compañía se integró con otras para formar después lo que en la actualidad es la International Business Machines Corporation (IBM). Otra persona, James Powers, también relacionado con el censo, diseñó equipo para procesar tarjetas con algunas características diferentes. Este equipo fue el que se utilizó para tabular el censo de 1910. En 1911 Powers formó la Powers Accounting Machine Company, adquirida posteriormente por Remington Rand (después división de Sperry Rand Corporation). Las tarjetas diseñadas por Hollerith eran de 80 columnas y perforaciones rectangulares, mientras que las diseñadas por Powers utilizaban 90 columnas y perforaciones circulares.

En 1937 Howard Aiken, de la Universidad de Harvard, diseñó una máquina que podía ejecutar automáticamente una secuencia de operaciones aritméticas. Al ser terminada en el año

de 1944 la Mark I era un calculador mecánico gigantesco. Contenía 72 acumuladores para sumar y 60 juegos de interruptores para poner constantes y sus instrucciones eran por medio de interruptores, botones, tableros con alambrado y cinta perforada. La información se representaba por patrones de relays mecánicos abiertos y cerrados. La máquina Harvard Mark I fue la predecesora inmediata de las computadoras automáticas.

Le siguió el ENIAC, que fue diseñado por J. Presper Eckert y John W. Mauchly, de la Moore School of Engineering de la Universidad de Pennsylvania, y lo terminaron en 1945. El ENIAC utilizaba componentes electrónicos y fue mucho más rápido que el Mark I. No tenía memoria de almacenamiento y se programaba por medio de interruptores y conexiones de alambres. El ENIAC se usó principalmente para el cálculo de tablas. Se dice que fue la primera computadora, aunque realmente fue una máquina de transición.

Después diseñaron otra máquina más grande llamada EDVAC, que era diferente del ENIAC en dos aspectos: utilizaba números binarios para operaciones aritméticas y almacenaba instrucciones en forma digital. El EDVAC se terminó de construir en 1952. En Inglaterra se terminó de construir antes el EDSAC y se considera como la primera máquina con memoria.

John Von Neumann, un matemático notable, se familiarizó con el trabajo de Eckert y Mauchly durante su estancia en la Moore School. Se interesó en el uso de las computadoras para análisis numérico así como en el diseño de ellas. Los escritos de Von Neumann son significativos debido a que contienen una descripción de la lógica de la computadora desarrollada por Eckert y otros en la Moore School. El computador IAS, que comenzó como proyecto del Princeton's Institute for Advanced Study y la Moore School, se describió en un artículo, indicando todos los dispositivos que se deben incorporar a una computadora. La computadora propuesta era una máquina paralela. La computadora IAS fue concluida en 1952, usó el sistema binario

y aritmética paralela. El EDVAC y el IAS introdujeron los diseños básicos para dos importantes tipos de computadoras: en serie y en paralelo. Las diferencias entre los dos tipos estriban en los métodos de mover la información de una parte de la máquina a otra y en la manera en que se lleva a cabo la suma. La máquina paralela mueve todos los dígitos que forman un número al mismo tiempo y la máquina en serie mueve un dígito a la vez. Cuando se usa el concepto en paralelo para la suma, todos los correspondientes pares de dígitos se suman simultáneamente. En una máquina en serie los pares de dígitos se suman un par a la vez, casi de la misma manera que la aritmética manual.

El UNIVAC es importante porque fue la primera computadora comercialmente disponible. Fue construida como una aventura comercial por la empresa Eckert Mauchly Computer Company, fundada en 1946 por J. Presper Eckert y J. W. Mauchly y adquirida en 1949 por Remington Rand. El UNIVAC I siguió al EDVAC en diseño, pero usando la estructura de una dirección del IAS.

La primera computadora UNIVAC entró en operación en la oficina de censos en abril de 1951. La primera instalación para un negocio fue en General Electric Appliance Park, en Louisville, Kentucky, en 1954. El nombre de UNIVAC fue sinónimo de "computadora" durante unos cuantos años hasta que IBM, que al principio sólo mostró un interés limitado en las computadoras, cambió de dirección y entró en este campo.

Las computadoras mencionadas fueron importantes porque representaron nuevos conceptos de diseño. Sin embargo, muchas otras computadoras se desarrollaron durante este periodo. El primero en operación fue el ENIAC en 1947, luego el EDSAC y el BINAC en 1949, cuatro máquinas separadas en 1950 y seis máquinas en 1951, año en el cual estuvieron disponibles comercialmente con dos UNIVACS instaladas.

En 1953 IBM instaló su primera computadora, la IBM 701. Un año después instaló la 650. La IBM 650 fue la computadora

más popular durante los siguientes cinco años. A todas estas computadoras se les conoce como computadoras de primera generación porque utilizaban bulbos de vacío; eran grandes, requerían mucho aire acondicionado y eran relativamente lentas. El transistor lleva a cabo la misma función que un tubo de vacío, pero es más pequeño, casi no genera calor y requiere muy poca energía, por lo que resulta muy barato. El cambio de bulbos a transistores comenzó con las computadoras militares en 1956 y con las comerciales en 1959. Las computadoras del periodo de 1959 a 1965 son conocidas como computadoras de segunda generación. Después de la aplicación de los transistores siguió una tendencia a la miniaturización o microelectrónica. Estas técnicas han obtenido componentes cada vez más pequeños, rápidos y seguros.

La tercera generación de computadoras se caracteriza por utilizar circuitos integrados. Utilizan hardware y software, están orientadas hacia la comunicación remota —programas internos, de control y ayudas de operación, datos por líneas telefónicas— además de efectuar operaciones simultáneas. Las computadoras de la tercera generación comenzaron a utilizarse en 1965. La más popular fue el sistema 360 de IBM.

A partir de 1970 se inició la tecnología de tipo monolítico. En un solo dado de silicio se integraban resistencias, capacitores, diodos y transistores. Cada vez los integrados requerían de menos energía y eran más baratos. Para 1982 IBM anunció un chip que podía almacenar 250 Kbits de información.

Actualmente, el hardware es cada vez más económico debido a las técnicas para fabricar chips de alta densidad. Las nuevas técnicas de integración evolucionan aceleradamente. Seguirá habiendo avances significativos en el área de procesamiento de datos gracias a la evolución del hardware.

1.4 Clasificación de los circuitos digitales

Los circuitos digitales se clasifican en *combinacionales* y *secuenciales*. Los *circuitos combinacionales* son aquellos en los que la salida en cualquier instante es una función exclusiva de la combinación de todas las entradas en dicho momento. Un comparador, un multiplexor, un decodificador son ejemplos de circuitos combinacionales. Los *circuitos secuenciales* son aquellos en los que la salida es función de las entradas y además de cierta secuencia de sucesos previos. Un contador es un ejemplo de un circuito secuencial.

La diferencia entre los circuitos secuenciales y los combinacionales es que los últimos no tienen retroalimentación. La mayoría de los circuitos secuenciales funcionan en sincronía con un pulso de reloj. La figura 1.4.1 muestra un diagrama de bloques de un circuito combinacional y la figura 1.4.2 la de un diagrama de un circuito secuencial.

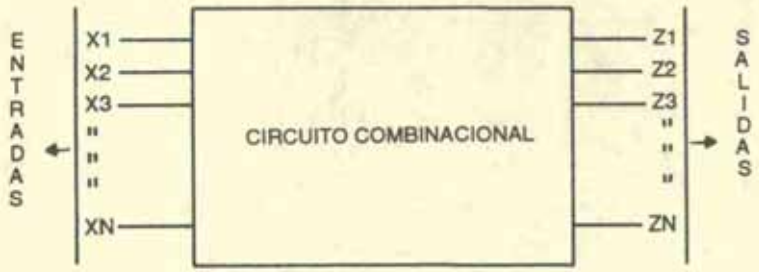


Figura 1.4.1. Circuito combinacional

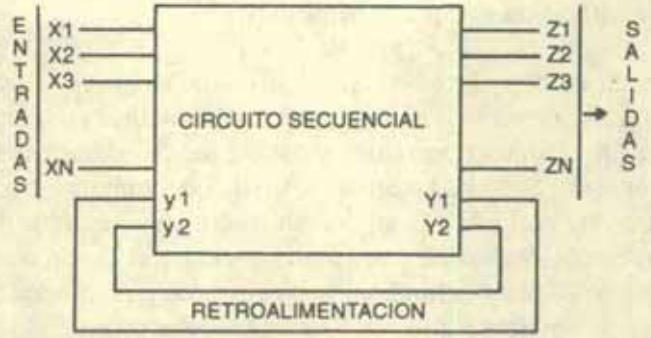


Figura 1.4.2. Circuito secuencial

1.5 Partes principales de una computadora

Una computadora es un sistema digital. Conviene estudiar su organización básica. La figura 1.5.1 muestra el diagrama de

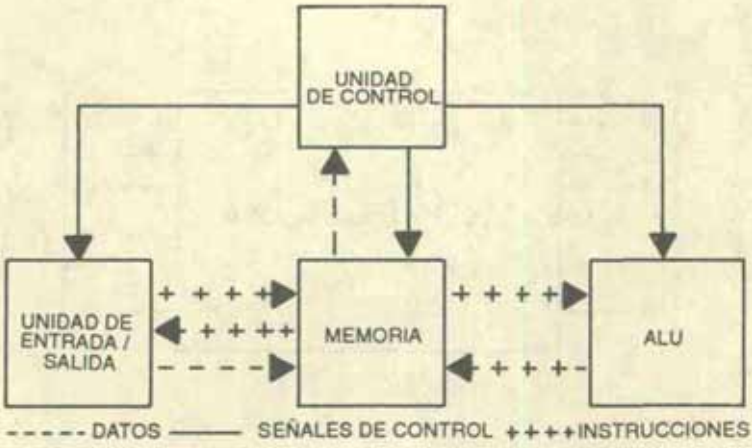


Figura 1.5.1. Organización básica de una computadora digital típica

bloques de una computadora digital. Consta de cuatro partes principales que se explican a continuación:

Unidad de control (CPU)

La unidad de control recibe las instrucciones para decodificarlas o interpretarlas. La unidad de control decide y clasifica estas instrucciones enviando las señales eléctricas a las otras partes de la computadora para que se realicen las operaciones apropiadas. Una instrucción típica puede ser que tome un número almacenado en cierta dirección de la memoria y lo sume al acumulador. La unidad de control enviará las señales adecuadas para hacer que el contenido en esa dirección se lea y se envíe a la unidad lógico-aritmética, y luego se sume al acumulador.

Otras instrucciones típicas pueden ser la lectura de un disco, el almacenamiento de datos en la memoria, la impresión de datos, la entrada en funcionamiento de un programa, etcétera.

Existen instrucciones de decisión o bifurcación. Las instrucciones que componen un programa de computadora se almacenan en orden en localidades secuenciales de la memoria. Por lo regular, la computadora realiza las instrucciones en secuencia. La orden de bifurcación le indica a la computadora que se desvíe a una localidad diferente, para obtener la siguiente instrucción, si se satisfacen ciertas condiciones. El CPU es un circuito secuencial.

Unidad de entrada o salida (I/O)

Un monitor, las unidades de disco, las impresoras, el teclado y todos los dispositivos que sirven de comunicación entre la computadora y el usuario son unidades de entrada o salida. Los datos y las instrucciones llegan a la computadora a través de la sección

de entrada y todos los resultados se devuelven a través de la sección de salida. Todos los datos y las instrucciones van directamente a la memoria, en donde se almacenan. La interfase está compuesta también por circuitos secuenciales.

Memoria

Se compone de dispositivos electrónicos. Dependiendo del tipo de éstos se clasifican en: memorias de sólo lectura (ROMs) y de lectura y escritura (RAMs). En capítulos posteriores se explicará su funcionamiento. Las memorias RAMs a su vez se clasifican en dinámicas y estáticas. En ellas se almacena la información, de tal manera que esté disponible para otras secciones de la computadora.

Toda la información que alimenta a la computadora se divide en unidades denominadas palabras o bytes, que es equivalente a 2 nibbles o a 8 bits. La memoria se divide en localidades individuales, cada una capaz de almacenar una sola palabra. Cada localidad tiene un número de identificación denominado dirección. La información almacenada en la memoria se compone de datos e instrucciones. Dentro de la máquina no hay ninguna diferencia de forma entre los dos, ya que ambos son simples colecciones de números o caracteres. La memoria ROM es un circuito combinacional y la memoria RAM un circuito secuencial.

Unidad lógica-aritmética (ALU)

Como ya se indicó, las instrucciones por lo regular se procesan en el CPU. Los datos se envían a la unidad lógica aritmética. El ALU se forma por registros, unidades electrónicas para el almacenamiento temporal de la información y ciertos circuitos lógicos que efectúan operaciones basadas en los números almacenados

en los registros. Frecuentemente existe un registro especial denominado acumulador, que se conecta permanentemente a los circuitos sumadores. La operación aritmética básica de la computadora consiste en sumar un número tomado de la memoria, a cualquier cifra que ya esté en el acumulador. El ALU que se utiliza es un circuito combinacional

1.6 Conclusiones

Actualmente los sistemas digitales se aplican en todos los campos de la tecnología. En los siguientes capítulos se mostrarán técnicas digitales para resolver problemas en algunos campos de la ingeniería, teniendo en consideración que el lector ya sabe dónde y cuándo se utilizan.

1.7 Ejercicios

1. Describir dos sistemas analógicos.
2. Describir dos sistemas digitales.
3. Dar tres ejemplos de circuitos combinacionales.
4. Dar tres ejemplos de circuitos secuenciales.
5. Expresar la diferencia entre un circuito secuencial y uno combinacional.
6. Indicar las ventajas y desventajas de un circuito en serie y uno en paralelo.

CAPÍTULO II
Algebra de Boole

Uno de los objetivos de este libro es demostrar que la teoría y la realidad van juntas. Por eso, la primera pregunta que se responderá es cuál es la aplicación de los circuitos digitales. Para encontrar la respuesta es necesario conocer algunas herramientas básicas de trabajo que se verán en este capítulo.

La utilidad de un sistema digital se mide por su capacidad para: a) tomar decisiones, y b) almacenar información.

En los sistemas digitales, cuando se quiere resolver una necesidad, se plantea el problema y luego se efectúa la implementación física de los procesos de decisión lógica.

Para entender estos conceptos se diseñará un sistema digital cuya función será activar una alarma. El planteamiento del problema es el siguiente:

Se debe instalar una alarma contra ladrones en un banco. Esta funcionará sólo si se activa el conmutador maestro en la estación de policía. Una vez activado el conmutador maestro, la alarma sonará si se cumple una de las siguientes condiciones: que la puerta de la bóveda se abra por la fuerza o que la puerta principal del banco sea abierta sin que el velador haya desactivado los dos interruptores anteriores.

Para manipular las proposiciones del problema anterior los conectivos binarios serán las herramientas que se necesitarán.

2.1 Conectivos binarios

El problema de la alarma está formado por algunas proposiciones:

A = La alarma suena,
C = Se activa el conmutador maestro,
B = La puerta de la bóveda es forzada,
P = La puerta principal del banco se abre,
G = El velador no desactiva los dos interruptores.

A cada proposición se le ha asignado una *variable de la proposición* que puede asumir un *valor de verdad*. Este valor de verdad puede ser verdadero o falso (V o F).

Una *proposición* es cualquier oración declarativa que pueda clasificarse como verdadera o falsa. A la parte de las matemáticas que manipula las variables de proposición y designa los valores de verdad se le llama *cálculo funcional de verdad*.

Hay proposiciones que niegan. Por ejemplo:

G = El velador no desactiva los dos interruptores

$\bar{G}$ es la negación de G. Otros símbolos para negar una proposición son ~G o G'.

Cuando una proposición está compuesta por dos o más proposiciones se le llama *compuesto funcional de verdad*. Su valor se puede determinar a partir de los valores de verdad de las proposiciones componentes. En el ejemplo se menciona:

La puerta de la bóveda es forzada
o
La puerta principal del banco se abre

La letra *o* conecta ambas proposiciones. Esto se puede expresar utilizando las variables de cada proposición y asignándole un signo al conectivo *o*.

$$B \vee P$$

Al conectivo $\vee$ se le denomina *OR inclusivo*, aunque por ser el más común nos referiremos a él simplemente como *OR*. Por lo tanto, las dos proposiciones B y P relacionadas por OR se pueden representar como $B \vee P$. Sólo existen cuatro combinaciones posibles de valores de verdad para las dos proposiciones componentes B y P que se definen por medio de una tabla que se llama *tabla de verdad*. La tabla de verdad define los compuestos funcionales de verdad. Analícese el compuesto funcional de verdad $B \vee P$. $B \vee P$ simboliza una proposición que es verdad si y sólo si B es verdad, o P es verdad, o los dos son verdad.

Cuando se tiene un compuesto funcional de verdad conectado con la letra *y* como:

el bebé come *y* el bebé es feliz,

se puede asignar a "el bebé come" la variable A y a "el bebé es feliz" la variable B. El conectivo que representa a **y** es $\wedge$. La proposición será $A \wedge B$ y con ella se simboliza una proposición que es verdad si y sólo si A es verdad y B es verdad. $\wedge$ es el conectivo de *y*. Se hará referencia a él por lo regular como el conectivo *AND*.

Se tiene también el conectivo OR exclusivo, y se simboliza por $A \oplus B$.

Como ejemplo se tiene el siguiente compuesto funcional de verdad con este conectivo:

José conducirá un trailer
o
un camión de volteo

Evidentemente, José no puede conducir ambos vehículos al mismo tiempo. Se utilizará entonces el conectivo $\oplus$, en el que $A \oplus B$ es verdad si y sólo si A es verdad o B es verdad, y será falsa cuando ambas proposiciones sean falsas o verdaderas. Otros símbolos del OR exclusivo son XOR u OE.

Otro conectivo es $\equiv$. Cuando se expresa $A \equiv B$, se quiere indicar que A y B siempre tienen el mismo valor de verdad. Los

A	B	$A \wedge B$	$A \vee B$	A XOR B	$A \equiv B$
F	F	F	F	F	V
F	V	F	V	V	F
V	F	F	V	V	F
V	V	V	V	F	V

Tabla 2.1.1

conectivos anteriores se tabularán en la siguiente tabla:
El problema planteado al inicio se puede expresar en términos del cálculo funcional de verdad como sigue: se deberá instalar una alarma contra ladrones en un banco. La alarma funcionará "A", *si y sólo si* se activa el conmutador maestro en la estación de policía "C" y (AND) la puerta de la bóveda es forzada "B" o

(OR) se abre la puerta principal del banco "P" y (AND) el velador no desactiva los dos interruptores anteriores G. Lo anterior puede expresarse como sigue:

$$A \equiv C \wedge ((B \vee P) \wedge G')$$

Hay 2^4, esto es, 16 formas de conectar dos variables, que se muestran a continuación:

		∧		A			B	OE	∨	↓	≡	B'		A'	⊃	↑	
A	**B**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
F	F	F	F	F	F	F	F	F	F	V	V	V	V	V	V	V	V
F	V	F	F	F	F	V	V	V	V	F	F	F	F	V	V	V	V
V	F	F	F	V	V	F	F	V	V	F	F	V	V	F	F	V	V
V	V	F	V	F	V	F	V	F	V	F	V	F	V	F	V	F	V

Tabla 2.1.2

En esta figura se tienen algunos conectivos nuevos. Uno de ellos es el siguiente:

$$A \supset B$$

"Si A es verdad, entonces B es verdad".

Esto significa que si A es verdad, B también va a ser verdad; pero si B es verdad, A no va a ser verdad por necesidad. Por ejemplo:

$$(2 > X > 1) \supset (X > 0)$$

Si, por ejemplo, X = 0.5, la proposición X > 0 es verdad pero la proposición 2 > X > 1 es falsa. Otro ejemplo:

"Si X es un número real ⊃ X es un número complejo".

Ayudándose de la tabla 2.1.2, se llegará a demostrar que:

$$(A \supset B) \wedge (B \supset A) \quad \equiv \quad (B \equiv A)$$

		PROPOSICIÓN 1				PROPOSICIÓN 2
A	B	(A ⊃ B)	∧	(B ⊃ A)	≡	(B ≡ A)
F	F	V	V	V		V
F	V	V	F	F		F
V	F	F	F	V		F
V	V	V	V	V		V

*1

*1 Ambos resultados son iguales. Se demuestra así que el valor de verdad de la proposición 1 es igual al valor de verdad de la proposición 2.

Tabla 2.1.3

Cuando se plantea un problema en cálculo funcional de verdad el diseñador tiene una responsabilidad doble, pues deberá ser muy preciso en la redacción de las especificaciones del problema. También se deberá tener cuidado al interpretar las especificaciones escritas por otras personas. Por ejemplo, una especificación puede ser la siguiente: "Si el interruptor está cerrado la

salida del circuito será de 5 voltios". Lo que se desea expresar es: "La salida del circuito será de 5 voltios si y sólo si el interruptor está cerrado".

Otros dos conectivos de gran interés son los que aparecen en la tabla 2.1.2:

$$A \uparrow B = \overline{A \wedge B}$$
$$A \downarrow B = \overline{A \vee B}$$

$A \uparrow B$ es la negación de la relación AND. Sus valores de verdad son precisamente opuestos a los de $A \wedge B$. Se le conoce como *NAND*. $A \downarrow B$ es la negación de $A \vee B$. Sus valores de verdad son precisamente opuestos a los del conectivo OR. Se le conoce como *NOR*. En seguida se verá cómo evaluar las funciones de verdad.

Cuando se quiere encontrar el valor de verdad de un compuesto se sustituyen las proposiciones por el valor de verdad y a continuación se evalúan. Véase un ejemplo:

Se quiere encontrar el valor de verdad de la siguiente proposición para el caso en el que tanto A como B sean falsas.

$$A' \vee (A \supset B)$$

Solución: A = F, B = F. V es el valor de verdad.

Sustituimos por sus valores de verdad:	A'	$\vee$	(A	$\supset$	B)
	V	V	F	V	F

El valor de verdad que resulta de la evaluación es verdadero. Primero se sustituye cada variable por su valor de verdad y luego se evalúan los compuestos funcionales de verdad, respetando

las prioridades definidas por los agrupamientos de paréntesis. En el ejemplo mostrado, primero se evaluó (A ⊃ B) yluego se obtuvo la resultante V. Como paso siguiente se conectó este valor de verdad con el valor de verdad de A, que es V. Como el conectivo es OR, el valor de verdad que resulta de V ∨ V, es V, que fue el resultado final de la evaluación.

Evalúese el mismo compuesto funcional de verdad mencionado anteriormente, con todos sus valores de verdad posibles:

A	B	A'	A⊃B	A'∨(A⊃B)	A'∨B	A'∨A	(A'∨A)⊃B
F	F	V	V	V	V	V	F
F	V	V	V	V	V	V	V
V	F	F	F	F	F	V	F
V	V	F	V	V	V	V	V

Tabla 2.1.4

Así es como se evalúan las proposiciones.

Es importante que en estas proposiciones no se confunda la expresión A' ∨ (A ⊃ B) con (A' ∨ A) ⊃ B. La colocación de los paréntesis es muy importante para su significado.

Cualquier expresión en función de sólo dos proposiciones A y B, relacionadas por cualquier número de conectivos binarios y donde los paréntesis designan un orden único de operación, puede reducirse a una lista de cuatro valores de verdad que corresponderán a cada combinación de los valores de A y B vistos en la tabla 2.1.2. Véase un ejemplo que verifica lo anterior:

Demostrar que la evaluación de la expresión A'∧(A' ∨ B) es una de las combinaciones de la tabla 2.1.2.

A B	A'	A'∨B	A'∧ (A'∨B)
F F	V	V	V
F V	V	V	V
V F	F	F	F
V V	F	V	F

Tabla 2.1.5

Como se habrá notado, el resultado total de la evaluación es igual a A', que es una de las combinaciones posibles de la tabla 2.1.2.

También puede haber compuestos como resultado de una suma de proposiciones, como sucede en el caso del problema planteado anteriormente. Considérese el compuesto funcional de verdad F, representado por:

$$F \equiv (A \vee B) \wedge (A' \vee C)'$$

En este caso se requiere una tabla de verdad de 2^3, esto es, de ocho renglones para obtener todas las combinaciones posibles para presentar las tres variables posibles de proposición. De esto se deduce que si el número de variables es n, habrá 2^n combinaciones posibles de valores de verdad. La tabla para evaluar el compuesto es la siguiente:

A B C	A'	(A∨ B)	(A' ∨ C)	(A'∨C)'	F
F F F	V	F	V	F	F
F F V	V	F	V	F	F
F V F	V	V	V	F	F
F V V	V	V	V	F	F
V F F	F	V	F	V	V
V F V	F	V	V	F	F
V V F	F	V	F	V	V
V V V	F	V	V	F	F

Tabla 2.1.6.

Otro detalle importante es que cualquier compuesto funcional de verdad que conste de cualquier número de proposiciones se puede expresar en función de ∨ , de ∧ y de NOT. Por ejemplo:

$A \supset B \equiv A' \vee B$. Esto se puede demostrar al evaluar la función. Si se desea, desarróllese como ejercicio.

Otro problema que ya se puede evaluar con las herramientas que se acaban de ver es el planteado al inicio de este capítulo sobre la alarma contra ladrones de banco.

2.2 Compuertas básicas

Los circuitos lógicos se relacionan directamente con el cálculo funcional de verdad. En la práctica, los únicos conectivos que existen son las compuertas AND, OR, NAND, NOR, OE u OR exclusivo y el inversor. La compuerta lógica es un dispositivo electrónico que cuenta con una terminal de salida y un número arbitrario de entradas. El potencial de voltaje con respecto a tierra de cualquier terminal de entrada o salida puede asumir uno de sólo dos valores específicos. Uno de los voltajes a través del sistema representa la verdad y el otro representa lo falso. En una compuerta AND la salida estará al voltaje que representa verdad si todas las entradas están en el voltaje determinado y se encontrará en el voltaje que representa lo falso si cualquiera de las entradas está en otro voltaje. En la compuerta OR la salida será verdad si cualquiera de las entradas es verdad y será falsa si todas las entradas lo son también.

Es importante notar que para todas las compuertas los voltajes de entrada y salida corresponden a la tabla de verdad de los conectivos binarios. La representación gráfica de los circuitos lógicos se da en la figura 2.2.1. Los símbolos se ilustran

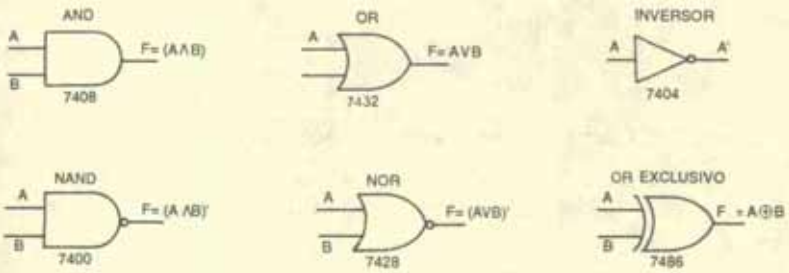


Figura 2.2.1. Compuertas básicas

con dos entradas. Sin embargo, las compuertas AND, NAND, OR y NOR pueden tener cualquier número de entradas. El OR exclusivo se define sólo para dos entradas.

La figura 2.2.7 muestra algunas compuertas básicas y su circuito equivalente. La figura 2.2.8 muestra el circuito equivalente de un inversor implementado con un relevador. El relevador tiene un interruptor NO (normalmente abierto) y otro NC (normalmente cerrado). Cuando el interruptor *a* está abierto, su LED estará prendido. Esto es, que cuando a = 0, el LED funcionará de acuerdo con a'=1. Cuando el interruptor *a* se encuentra cerrado se crea un campo magnético en la bobina, causando que el interruptor que estaba normalmente cerrado se abra, al tiempo que el LED se apaga. La figura 2.2.2 presenta un ejemplo de una aplicación más compleja de ésto:

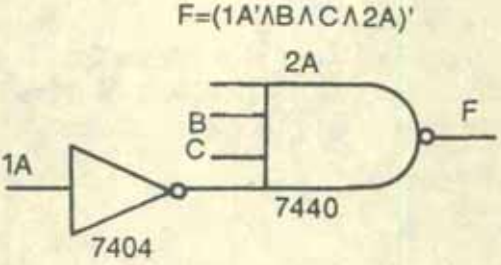


Figura 2.2.2. Compuerta NAND con inversor

Si se vuelve hacia el problema inicial de la alarma de banco, resulta que:

A = la alarma suena,
C = se activa el conmutador maestro,
B = la puerta de la bóveda es forzada,

P = la puerta principal del banco se abre, y
G' = el velador no desactiva los dos interruptores.

La relación de los enunciados anteriores se representa de la siguiente manera:

$$A \equiv C \wedge (G' \wedge (B \vee P))$$

El circuito digital se muestra en la figura 2.2.3.

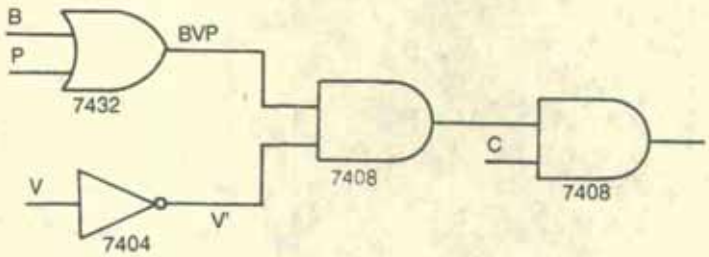


Figura 2.2.3

Procedimiento para el diseño de los circuitos

Paso 1. Obtener una proposición del problema de diseño que se pueda simbolizar como un compuesto funcional de verdad.
Paso 2. Obtener una expresión de la salida del problema en función de los conectivos AND, OR, NAND, NOR, NOT y OE.
Paso 3.- Obtener la expresión lógica.

Paso 4. Construir la realización física.

Es importante saber que cualquier compuesto funcional de verdad se puede expresar en función de AND, OR y NOT. Cualquier conectivo se puede sustituir por cualquier combinación de los tres anteriores. Esto se demuestra en el siguiente listado:

$f_0 = A \wedge A'$
$f_1 = A \wedge B$
$f_2 = A \wedge B'$
$f_3 = A$
$f_4 = A' \wedge B$
$f_5 = B$
$f_6 = (A' \wedge B) \vee (A \wedge B')$
$f_7 = A \vee B$
$f_8 = (A \vee B)' = A' \wedge B'$
$f_9 = (A' \wedge B') \vee (A \wedge B)$
$f_{10} = B'$
$f_{11} = A \vee B'$
$f_{12} = A'$
$f_{13} = A' \vee B$
$f_{14} = (A \wedge B)' = A' \vee B'$
$f_{15} = A \vee A'$

Hay dos teoremas importantes, los teoremas de De Morgan. Son los siguientes:

1) $A \downarrow B = (A \vee B)' = A' \wedge B'$

2) $A \uparrow B = (A \wedge B)' = A' \vee B'$

Ejercicio. Evaluar los teoremas de De Morgan y demostrar su veracidad al comparar los resultados de las tablas de verdad. La tabla 2.2.1 muestra que la evaluación de A' ∨

B' es igual a (A ∧ B)', y A' ∧ B' es igual a (A ∨ B)'.

Otro aspecto importante es que cualquier compuesto fun-

A B	A' B'	A' ∨ B'	(A∧ B)'	A' ∧ B'	(A ∨ B)'
F F	V V	V	V	V	V
F V	V F	V	V	F	F
V F	F V	V	V	F	F
V V	F F	F	F	F	F

Tabla 2.2.1. Demostración de los teoremas de De Morgan

cional de verdad puede expresarse en función de un solo conectivo, que puede ser ↑ o ↓. Véanse los siguientes ejemplos en los que se expresan diferentes funciones con compuertas NAND solamente, y luego con compuertas NOR.

Ejemplo 1

X' = (X ∧ X)' = X ↑ X

Ejemplo 2

X ∧ Y = (X ∧ Y)' ' = (X ↑ Y)' =
((X ↑ Y) ∧ (X ↑ Y))' =
((X ↑ Y) ↑ (X ↑ Y))

Ejemplo 3

X ∨ Y = (X ∨ Y)' ' = (X' ∧ Y')' =
(X ↑ X) ↑ (Y ↑ Y)

Ejemplo 4

X' = (X ∨ X)' = X ↓ X

Ejemplo 5

$$X \wedge Y = (X \wedge Y)'' = (X' \vee Y')' = (X \downarrow X) \downarrow (Y \downarrow Y)$$

Ejemplo 6

$$X \vee Y = (X \vee Y)'' = (X \downarrow Y)' = ((X \downarrow Y) \vee (X \downarrow Y))' = (X \downarrow Y) \downarrow (X \downarrow Y)$$

Ejemplo 7

$$A' \vee B = (A' \vee B)'' = (A'' \uparrow B') = (A \uparrow A)' \uparrow (B \uparrow B) = A' \vee B = ((A \uparrow A) \uparrow (A \uparrow A)) \uparrow (B \uparrow B)$$

Este concepto es importante porque indica que cualquier circuito lógico puede construirse utilizando sólo compuertas NOR o NAND.

Ejemplo 8

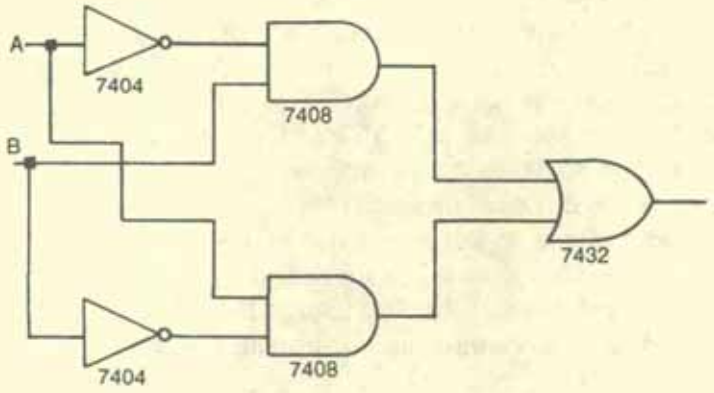
Convertir un OE en función de sólo compuertas NAND.

$$X \text{ OE } Y = (A \wedge B') \vee (A' \wedge B) = ((A \wedge B') \vee (A' \wedge B))'' = ((A \wedge B')' \wedge (A' \wedge B')')' = (A \uparrow B') \uparrow (A' \uparrow B) = (A \uparrow (B \uparrow B)) \uparrow ((A \uparrow A) \uparrow B)$$

El circuito de la compuerta OR exclusivo se muestra en la figura 2.2.4, así como su circuito equivalente en función exclusiva de compuertas NAND.

Ejercicio. Convertir un OE en función de compuertas NOR exclusivamente.

Todos estos circuitos se pueden implementar con circuitos integrados TTL. La numeración de los circuitos TTL por lo regular es de 74xx, donde el valor de la x dependerá de la configuración del circuito. En las figuras 2.2.1 a 2.2.4 los circuitos traen su número correspondiente en TTL. Cuando las compuertas son de



F=(A∧B)∨(A'∧B)=((A∧(B∧B)')'∧((A∧A)'∧B)')'

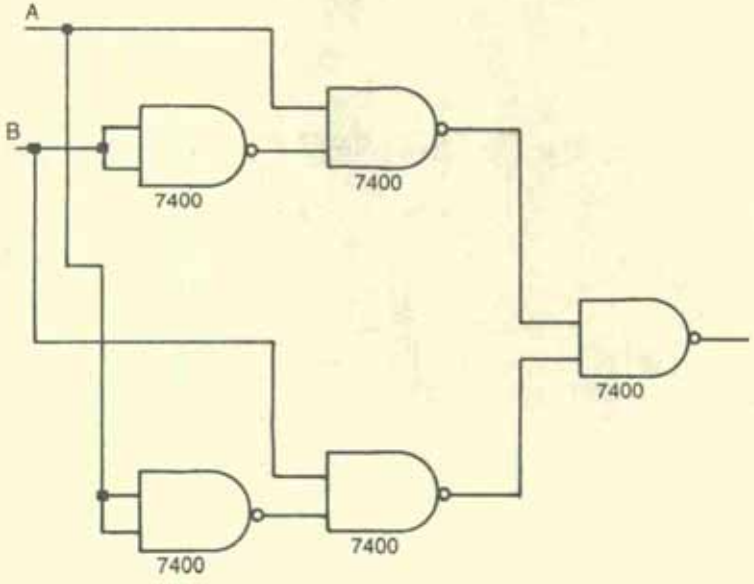


Figura 2.2.4. Todo circuito lógico se puede construir utilizando sólo compuertas NOR o NAND

dos entradas, por lo regular el circuito integrado trae integradas en un solo paquete cuatro compuertas. Se recomienda ver el manual de circuitos TTL para ver las especificaciones del circuito, así como la ubicación de entradas y salidas de compuertas, la alimentación y la tierra. Se recomienda trabajar con la familia de ICs (Circuitos Integrados) 74LSXX por consumir menos potencia. Más adelante se explicarán las características principales de cada una de estas familias de ICs.

Otro detalle importante es que en los ICs TTL las entradas de compuertas que se dejan sin conectar representan una entrada verdadera en el circuito. La figura 2.2.5 muestra la configuración interna de un IC que tiene cuatro compuertas NOR de dos entradas.

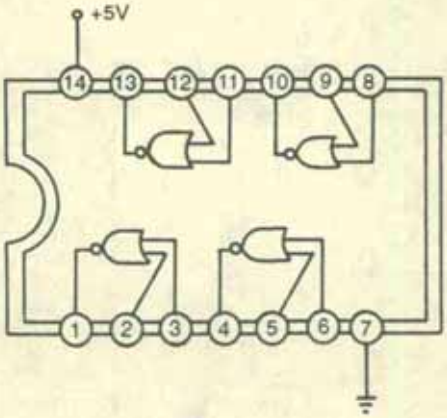


Figura 2.2.5. Ic con compuertas tipo NOR

La figura 2.2.6 muestra cómo se implementaría la función:

$$F = (A' \wedge B) \vee (C \wedge D).$$

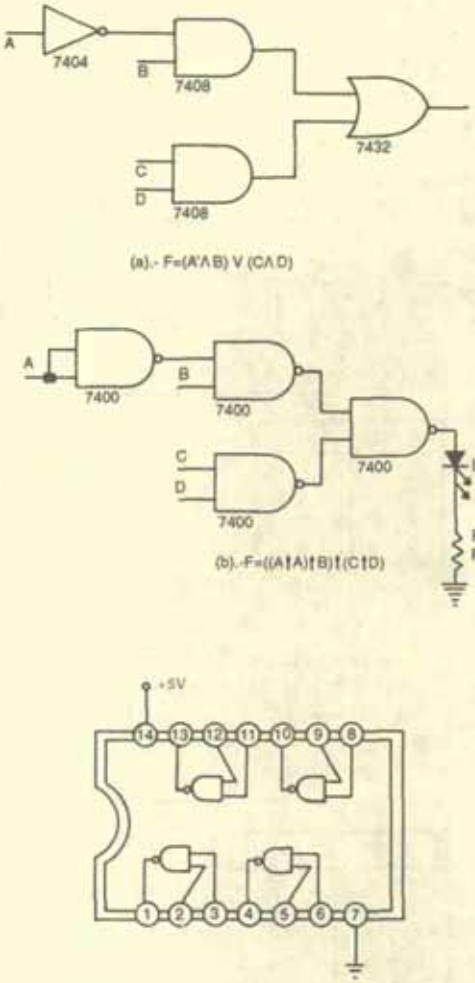
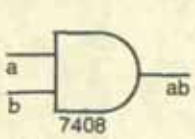
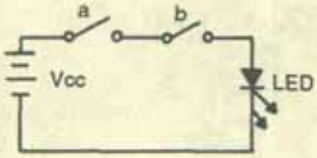


Figura 2.2.6. Las figuras muestran dos opciones para implementar el mismo circuito y el CI requerido para la segunda opción

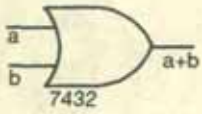
Compuerta AND de dos entradas

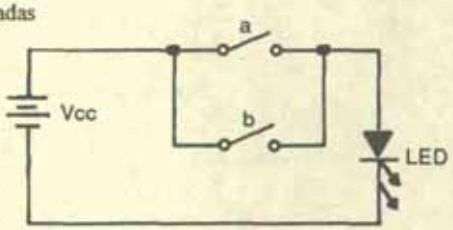




a	b	a
F	F	F
F	V	F
V	F	F
V	V	V

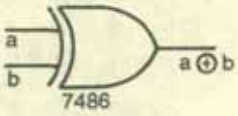
Compuerta OR de dos entradas

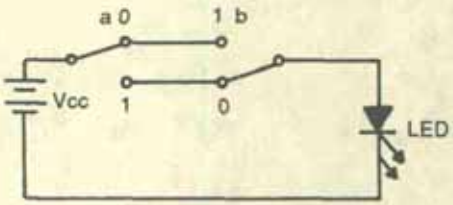




a	b	a+b
F	F	F
F	V	V
V	F	V
V	V	V

Compuerta XOR





a	b	a⊕b
F	F	F
F	V	V
V	F	V
V	V	F

Figura 2.2.7. Compuertas básicas y su circuito equivalente con interruptores

INVERSOR. Su circuito consta de un relevador donde se conecta al interruptor que está normalmente cerrado NC.

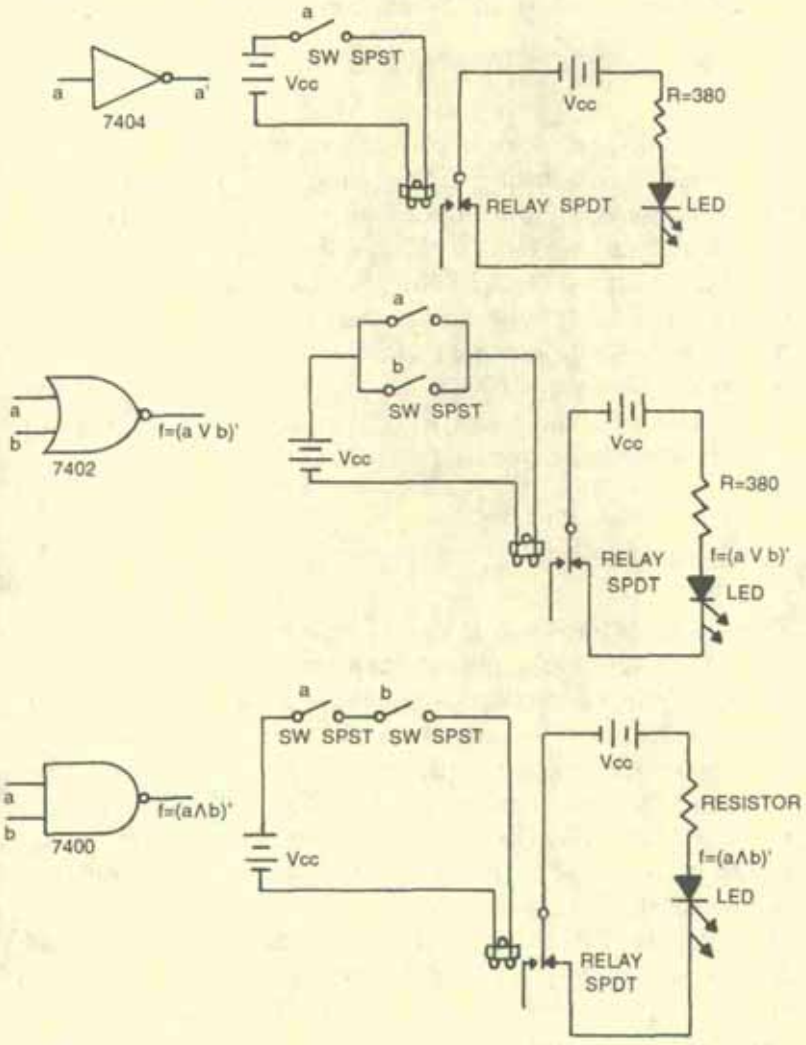


Figura 2.2.8. Compuertas básicas y su circuito equivalente

Primero detalla el circuito utilizando un inversor, dos compuertas AND y una compuerta OR. A continuación se muestra la ecuación del circuito armado sólo con compuertas tipo NAND:

$$F = ((A \uparrow A) \uparrow B) \uparrow (C \uparrow D).$$

Este ejemplo presenta con claridad por qué es tan importante la posibilidad de convertir un compuesto en función de sólo un tipo de compuerta. Para un circuito que utiliza compuertas NAND sólo tendrá que utilizarse un IC, que es precisamente el que se muestra en las figuras 2.2.6 (b) y (c). El circuito que se ilustra en la figura 2.2.6 (a) utilizará, en cambio, tres circuitos integrados: un inversor de seis que tiene el 7404, dos compuertas AND de cuatro que tiene el 7408 y una compuerta OR de cuatro que tiene el 7432. Como podrá apreciarse, con esta última opción muchas compuertas quedarían sin usarse.

2.3 Algebra de Boole

Técnicamente hablando, la lógica es una ciencia que trata sobre los criterios de validez utilizando los principios del razonamiento. En ingeniería, estos criterios se aplican a las funciones lógicas AND, OR, NAND y NOR, aunque la lógica en sí misma implica un concepto mucho más amplio que estas pocas funciones limitadas. Los conceptos más importantes de la lógica aplicada a la ingeniería los desarrolló Boole. A continuación se verán estas funciones con el detalle suficiente para que ayuden a simplificar problemas de circuitos lógicos.

El álgebra de Boole está basada en elementos que tienen dos estados estables posibles. Es por eso que es muy útil para analizar circuitos de conmutación. La razón de esto es que un interruptor o un circuito digital, pueden tener sólo dos estados posibles: abierto o cerrado, 0 ó 1, falso o verdadero. Se verán las

operaciones básicas de conmutación, pero hay que tener presente que todas las operaciones de conmutación, aún las más complejas, son meras combinaciones de estas operaciones básicas. A continuación se enlistarán las leyes fundamentales del álgebra de Boole, que deberán memorizarse puesto que se usarán para simplificar expresiones.

CÁLCULO DE VERDAD	ALGEBRA DE BOOLE
∧	•
∨	+
F	0
V	1
A'	A'

Tabla 2.3.1. Equivalencia de símbolos entre el cálculo funcional de la verdad y el álgebra de Boole

Véase ahora un ejemplo, en el que se evaluará una función utilizando símbolos del cálculo funcional de verdad y del álgebra de Boole:

CÁLCULO DE VERDAD				ALGEBRA DE BOOLE			
A	B	A∧B	A∨B	A	B	A • B	A + B
F	F	F	F	0	0	0	0
F	V	F	V	0	1	0	1
V	F	F	V	1	0	0	1
V	V	V	V	1	1	1	1

Tabla 2.3.2. Ejemplo de equivalencias entre cálculo funcional de la verdad y el álgebra de Boole

2.4 Leyes y axiomas fundamentales del álgebra de Boole

Son nueve las leyes del álgebra de Boole. Las más importantes son las tres primeras:

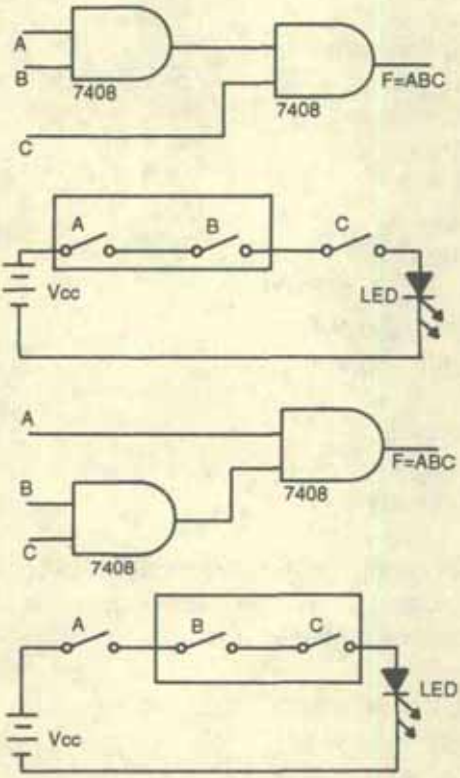
1. Ley asociativa
   (a) $(A \bullet B) \bullet C = A \bullet (B \bullet C)$
   (b) $(A + B) + C = A + (B + C)$
2. Ley conmutativa
   (a) $A \bullet B = B \bullet A$
   (b) $A + B = B + A$
3. Ley distributiva
   (a) $A \bullet (B + C) = A \bullet B + A \bullet C$
   (b) $A + (B \bullet C) = (A + B) \bullet (A + C)$

4. Ley de identidad
   $A = A$
5. Ley de complementariedad
   (a) $A \bullet A' = 0$
   (b) $A + A' = 1$
6. Ley de idempotencia
   (a) $A \bullet A = A$
   (b) $A + A = A$
7. Ley de dualización
   (a) $A + 1 = 1, A \bullet 0 = 0$
   (b) $A + 0 = A, A \bullet 1 = A$
8. Ley de la doble negación
   $A = A''$
9. Ley de absorción
   (a) $A \bullet (A + B) = A$
   (b) $A + (A \bullet B) = A$

Un teorema es una proposición que necesita ser demostrada. En seguida se verán tres teoremas fundamentales del álgebra de Boole, los teoremas de De Morgan:

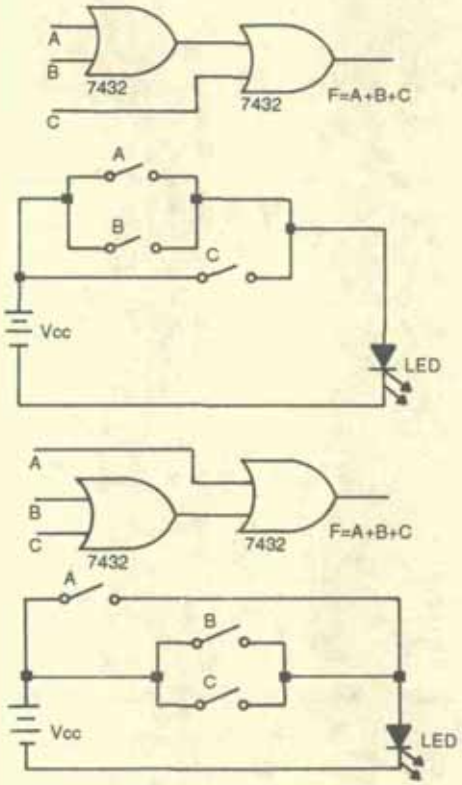
1. (a) $(A + B)' = A' \bullet B'$
   (b) $(A \bullet B)' = A' + B'$
2. (a) $A + (A' \bullet B) = A + B$
   (b) $A \bullet (A' + B) = A \bullet B$
3. $AB + A'C + BC = AB + A'C$

Las figuras 2.4.1, 2.4.2, 2.4.3, 2.4.4 y 2.4.5 son las demostraciones de las tres primeras leyes del álgebra de Boole. El resto de las nueve leyes y los tres teoremas de De Morgan se pueden demostrar de la misma manera. En realidad, algunas de estas leyes son evidentes por lo que no necesitan ser demostradas a partir de otras.



A	B	C	AB	(AB)C	BC	A(BC)
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

Figura 2.4.1. Ley asociativa
Demostración de (A • B) • C = A • (B • C)



A	B	C	A+B	(A+B)+C	B+C	A+(B+C)
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	1	1	0	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Figura 2.4.2. Ley asociativa
Demostración de (A + B) + C = A + (B + C)

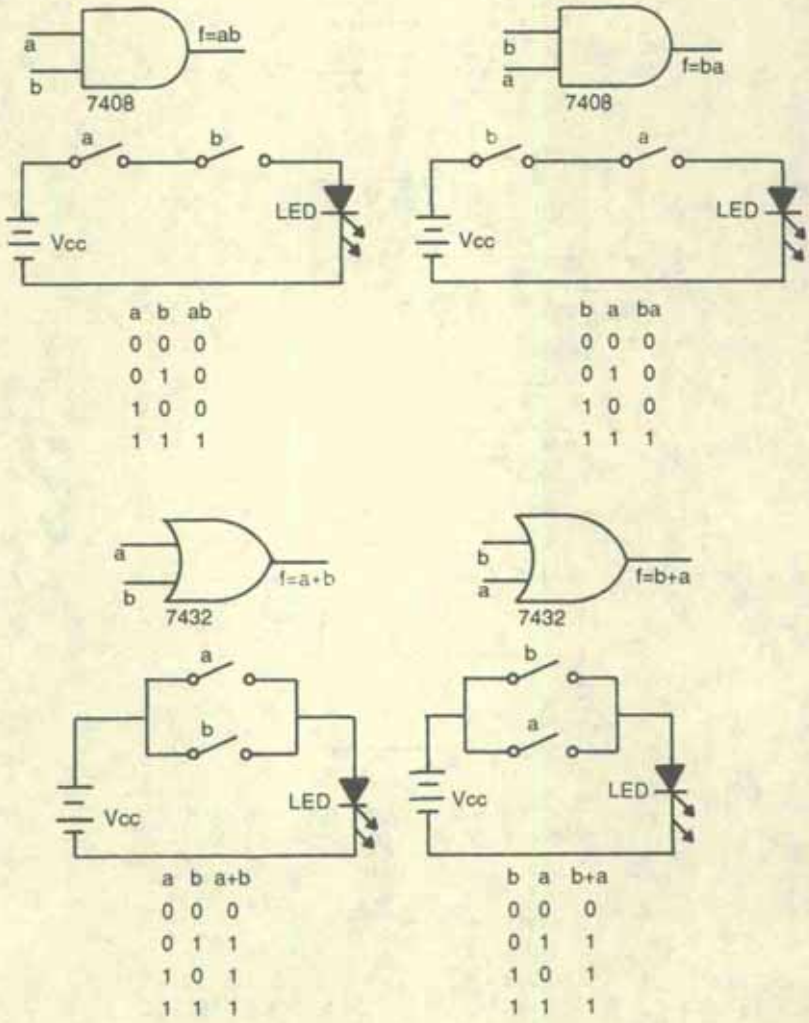


Figura 2.4.3. Ley conmutativa. Diagramas lógicos. Circuito con interruptores y tablas de verdad

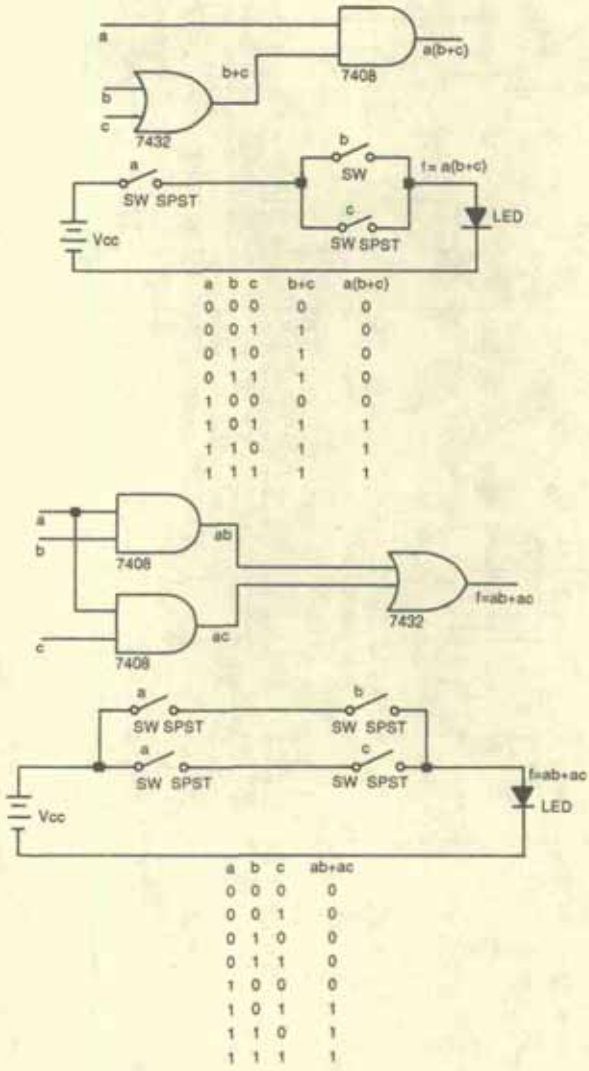
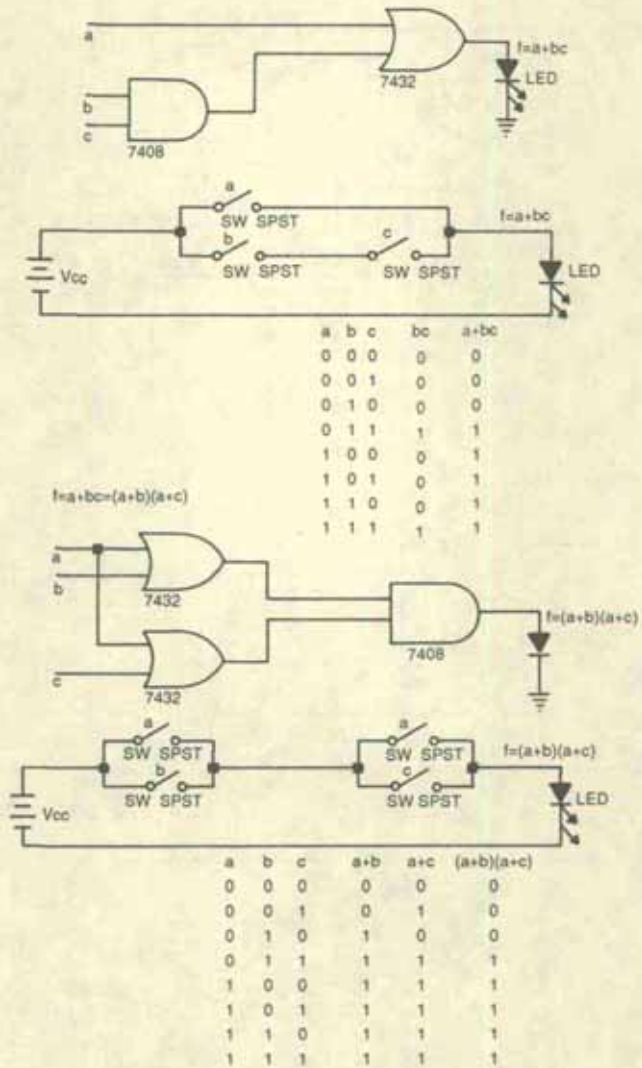


Figura 2.4.4. Ley distributiva.F = A • (B + C) = A • B + A • C. Circuito con interruptores y tablas de verdad



a	b	c	bc	a+bc
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

a	b	c	a+b	a+c	(a+b)(a+c)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

Figura 2.4.5. Ley distributiva. Diagramas lógicos. Circuito con interruptores y tablas de verdad

En las leyes y teoremas del álgebra de Boole existe simetría. Esto significa que para cada teorema acerca de productos existe también un teorema análogo acerca de sumas. Por ejemplo, en la ley de la absorción se encuentran las siguientes ecuaciones:

$$A\,(A + B) = A \quad Y \quad A + (A\,B) = A$$

Las ecuaciones se demostrarán más adelante. Ahora se verá que un elemento A puede ser multiplicado por (A + B) y el elemento no será afectado. Si se suma A B a A el resultado será todavía A. Esta propiedad recibe el nombre de *dualidad*: si hay un elemento que puede ser multiplicado por cualquier otro elemento y no ser afectado, entonces puede haber otro elemento que puede ser sumado y que tampoco afectará dicho elemento.

El *dual* de cualquier proposición se puede obtener sustituyendo las sumas por multiplicaciones, los unos por ceros, y viceversa. Así por ejemplo, el dual para la ley distributiva de:

$$a + (b \bullet c) = (a + b) \bullet (a + c)$$

será:

$$a \bullet (b + c) = (a \bullet b) + (a \bullet c)$$

A continuación se demuestran de manera analítica los teoremas y algunas de las leyes que la necesitan:

a) *Ley de idempotencia*: $A + A = A$ y $A \bullet A = A$

$A + A = (A + A) \bullet 1$; si $1 = A + A'$, entonces:

$= (A + A) \bullet (A + A')$; aplicando la ley distributiva:

$= A + (A \bullet A')$; si $A \bullet A' = 0$

$= A + 0$

$= A$

$A \bullet A = A$ dual.

b) $A + 1 = 1$ y $A \bullet 0 = 0$

$A + 1 = 1 \bullet (A + 1)$

$= (A + A') \bullet (A + 1)$; aplicando la ley distributiva:

$= A + A' \bullet 1$

$= A + A'$ si $A + A' = 1$, entonces:

$= 1$

$A \bullet 0 = 0$ dual.

c) *Ley de absorción*: $A \bullet (A + B) = A$ y $A + (A \bullet B) = A$

$A + AB = A \bullet 1 + A \bullet B$; aplicando la ley distributiva:

$= A \bullet (1 + B)$ pero $1 + B = 1$, entonces:

$= A \bullet 1$

$= A$

$A \bullet (A + B) = A$ dual.

d) *Teoremas de De Morgan*:

$(A + B)' = A' \bullet B'$ y $(A \bullet B)' = A' + B'$.

Si $(A + B)' = A' \bullet B'$ y si $A + A' = 1$, entonces:

$(A + B) + A' \bullet B' = 1$;aplicando la ley distributiva (demostración):

$= ((A + B) + A') \bullet ((A + B) + B')$

$= (A + B + A') \bullet (A + B + B')$

$= (A + A' + B) \bullet (A + B + B')$

$= (1 + B) \bullet (A + 1)$

$= 1 \bullet 1$

Se aplicaron las leyes asociativa y distributiva, por lo que se sabe que $1 + A = 1$.

e) Demostración de: $A + A'B = A + B$

Se tiene que $A + A' B = (A + A') (A + B)$ (ley distributiva)

$= 1 (A + B)$

$= A + B$

$A \bullet (A' + B) = A B$ dual.

f) Demostración de $AB + A'C + BC = AB + A'C$

$$\begin{aligned} AB + A'C + BC &= AB + A'C + BC(A + A') \\ &= AB + A'C + ABC + A'BC \\ &= AB + ABC + A'C + A'BC \\ &= AB(1 + C) + A'C(1 + B) \\ &= AB \bullet 1 + A'C \bullet 1 \\ &= AB + A'C. \end{aligned}$$

Las leyes y teoremas del álgebra de Boole servirán para minimizar funciones, pero no se obtendrá el resultado óptimo. Para ello se emplearán otros métodos.

2.5 Teoría de conjuntos

Tanto el cálculo funcional de verdad como la teoría de conjuntos satisfacen las leyes y los teoremas del álgebra de Boole. Por eso se dará un breve repaso de la teoría de conjuntos.

Un conjunto es cualquier colección de objetos. El conjunto universal contiene un número finito o infinito de objetos.

R es un subconjunto de S cuando cada objeto de R se encuentra también en S (Ver figura 2.5.1).

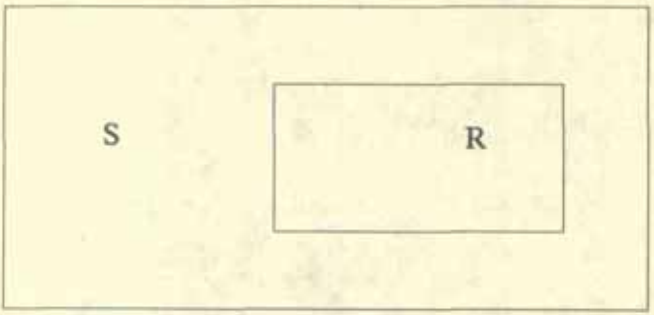


Figura 2.5.1. R es subconjunto de S. Se simboliza R § S

El conjunto vacío es un conjunto que no tiene ningún elemento. Su símbolo es S_Z. El conjunto universal está compuesto por todos los elementos. Su símbolo es S_U.

La *unión* de dos conjuntos A y B es el conjunto que contiene los objetos tanto de A como de B. Su símbolo es A ∪ B. La *intersección* de los conjuntos A y B es el conjunto que contiene los elementos que se encuentran en ambos conjuntos A y B. Si un elemento sólo pertenece a A o sólo pertenece a B, entonces no forma parte del conjunto de A intersección B. Su símbolo es A ∩ B. El complemento de un conjunto A es C(A), contiene todos los objetos que se encuentran en el conjunto universal pero no los que están en A.

Lo anterior se ilustra con los diagramas de Venn, como ya se vio en la figura 2.5.1. La figura 2.5.2 muestra los diagramas de Venn para la intersección, la unión y el complemento.

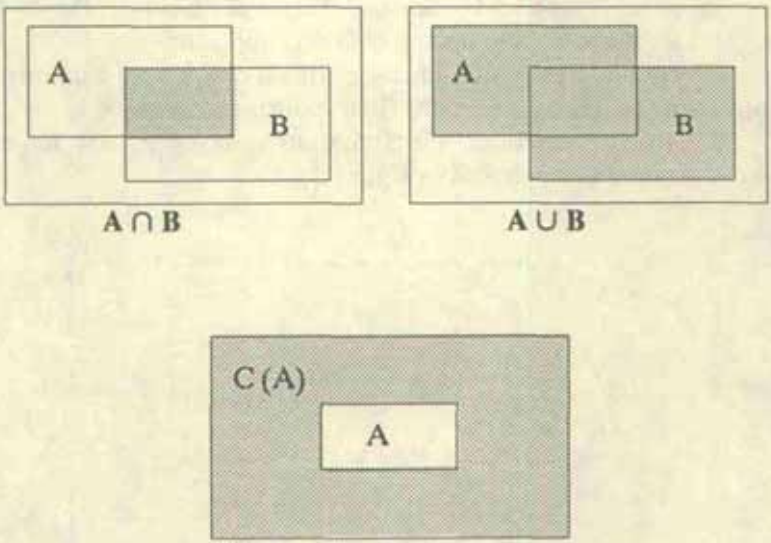


Figura 2.5.2. Diagramas de Venn que muestran A ∩ B, A ∪ B y C (A)

Correspondencia entre la teoría de conjuntos y el álgebra de Boole:

(intersección) $\cap$ = • (multiplicación)
(unión) $\cup$ = + (suma)
(conjunto vacío) S_Z = 0
(conjunto universal) S_u = 1
(complemento) C (S) = S'

La teoría de conjuntos también satisface las leyes y los teoremas del álgebra de Boole. Por ejemplo:

$$A \cup S_U = S_U, \text{ y}$$
$$A + 1 = 1$$

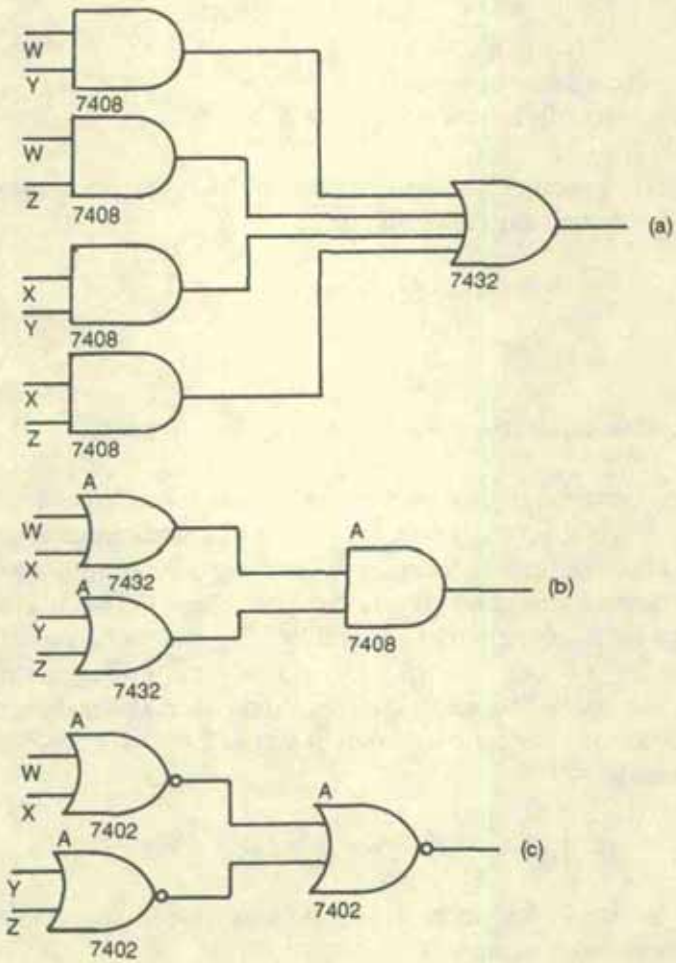
2.6 Aplicación de los teoremas del álgebra de Boole

Se han visto ya las leyes y teoremas que comprenden el álgebra booleana, que son muy útiles para describir ecuaciones lógicas, tales como las que se usan en la tecnología de computadoras. Estas leyes y teoremas son usados para simplificar ecuaciones lógicas de tal manera que los circuitos lógicos puedan ser diseñados en la forma más simple y económica. Por ejemplo, la ecuación que se muestra a continuación es una ecuación lógica que describe el circuito lógico de la figura 2.6.1 (a) en términos booleanos:

$$F(w, x, y, z) = wy + wz + xy + xz$$

Si se utiliza la ley de la distribución se puede simplificar la ecuación como sigue:

$$F = w(y + z) + x(y + z)$$
$$= (w + x)(y + z)$$



Figuras 2.6.1 (a), (b) y (c)

Se podrá apreciar que la ecuación anterior representa el circuito de la figura 2.6.1 (b), cuya función es la misma que la del circuito lógico 2.6.1 (a) pero con menos componentes. Así es como se ha simplificado el circuito. Todavía se puede hacer más. Si se niegan dos veces cada uno de los términos y luego se aplica el teorema de De Morgan, el circuito utilizará únicamente compuertas NOR, hecho que sería muy conveniente porque en vez de utilizar dos circuitos integrados, un 7408 para la compuerta AND y un 7432 para la compuerta OR, se utilizará sólo un 7402. Véase cómo se hace:

$$F = (w + x)'' (y + z)''$$; aplicando el teorema De Morgan:
$$= ((w + x)' + (y + z)')'$$

El circuito queda como se aparece en la figura 2.6.1 (c).

Considérese otro ejemplo en detalle. Supóngase que se quiere simplificar la siguiente ecuación lógica:

$$f = a\,b\,c + a\,b\,d' + a\,c' + a'\,b'\,c'\,d' + a'\,c$$

Paso 1. Se reacomodan los términos como sigue:
$$f = abc + ac' + a'b'c'd' + a'c + abd'$$
$$= a\,(b\,c + c') + a'\,(b'\,c'\,d' + c) + a\,b\,d'$$
Paso 2. Se aplica el teorema que dice $a + a'\,b = a + b$:
$$= a\,(b + c') + a'\,(b'\,d' + c) + a\,b\,d'$$
Paso 3. Se aplica la ley distributiva:
$$= a\,b + a\,c' + a'\,b'\,d' + a'\,c + a\,b\,d'$$
$$= (a\,b + a\,b\,d') + a\,c' + a'\,c + a'\,b'\,d'$$
Paso 4. Se aplica la ley de la absorción:
$$= a\,b + a\,c' + a'\,c + a'\,b'\,d'$$
Paso 5. Para finalizar se factoriza, quedando:
$$= a\,(b + c') + a'\,(c + b'\,d')$$

La mejor manera de familiarizarse con el uso de los teoremas y las leyes del álgebra de Boole es utilizarlos repetidas veces. Estos teoremas son muy valiosos en la simplificación de funciones elementales.

2.7 Conclusiones

El álgebra de Boole es importante porque juega un papel fundamental en el diseño de los circuitos digitales. Se recomienda que al terminar de leer este tema se realicen la prácticas 1 y 2. La práctica 1 consiste en comprobar las operaciones básicas del álgebra de Boole en los circuitos combinacionales más comunes, como son las compuertas. La práctica 2 requiere que se haga uso de estas herramientas para diseñar un circuito con una aplicación particular, como una alarma o un detector de números pares.

2.8 Ejercicios

Simplificar las siguientes funciones, dibujar el diagrama lógico y representar cada función sólo con compuertas tipo NAND:

1. f = a b' d + a b' d'
2. f = (a' + b) (a + b)
3. f = a c d + a' b c d
4. f = (a + b' + a b) (a' b) (a + b)
5. f = (a + b') (a b c + b' (c + a)) + a b c' (a + a' b)
6. f = (a + b' + a b') (a b + a' c + b c)
7. f = (c' + d) (c' + d + e) (a b + c + d)
8. f = (a c + a c' (b + b')) (a + a' b')
9. f = ((a + b)' ((a + (a' + c d) a') (d' + a')))'
10. f = (a'b' (ca + b')) + (a + b) ((ab'c') + (a'bc))' '

Respuestas:

1. f = a b'
2. f = b
3. f = a c d + b c d
4. f = 0
5. f = a + b' c
6. f = a b + a' b' c
7. f = a b c' + d
8. f = a
9. f = a + b
10. f = c a' + c' b'

CAPÍTULO III
Funcionamiento de los dispositivos en forma digital

En los capítulos anteriores se hizo referencia a las compuertas AND, OR, NAND, NOR, y cómo varios circuitos se podían combinar para obtener cierta salida deseada al aplicar ciertas entradas. A estos circuitos se les llama circuitos de compuerta.

En este capítulo se examinará el funcionamiento de este tipo de circuitos que pueden estar compuestos por transistores, diodos y otros componentes. En seguida se explicará el funcionamiento básico de los componentes electrónicos utilizados en estos circuitos.

3.1 Semiconductores

Antes de entrar en el estudio del diseño de circuitos digitales es necesario saber que todos ellos están hechos de cristales llamados semiconductores. Un semiconductor es conductor o aislante eléctrico, dependiendo de ciertas condiciones. Hay muchos materiales semiconductores, de los que el más popular es el silicio, ingrediente principal en la arena.

El silicio tiene cuatro electrones en la última órbita de su estructura química. Se ha de recordar que la última órbita de un

átomo de cualquier elemento tiene un máximo de ocho electrones. Por lo tanto, un átomo de silicio está listo para combinarse con otro átomo para compartir electrones, de manera que ambos cuenten con ocho electrones en su última órbita.

Cuando se combinan átomos de silicio forman un cristal. Los átomos de silicio que quedan en la superficie de la estructura cristalina serán los únicos que tendrán cuatro electrones en su última órbita. Los demás se combinarán entre sí de tal manera que compartan sus electrones teniendo así ocho en su última órbita. El silicio es muy abundante en nuestro planeta. Forma el 27.7% de la superficie de la Tierra. Sólo el oxígeno es más común. Nunca se encuentra en estado puro. Es de color gris obscuro cuando se purifica. El silicio y el diamante comparten la misma estructura cristalina además de otras propiedades, pero el silicio no es transparente. Con él se forman barras cilíndricas, como con el salami, que después se cortan en obleas para hacer partes electrónicas.

En electrónica, el silicio puro no es muy útil. Debe estar combinado con fósforo, boro u otros elementos. Cuando se combina adquiere propiedades muy importantes para su uso en electrónica. La razón es la siguiente. Un átomo de boro tiene sólo tres electrones en su última órbita, mientras que un átomo de fósforo tiene cinco. Si el silicio se combina con fósforo es llamado silicio con impurezas tipo N-negativo. En cambio, el silicio con boro se llama silicio con impurezas tipo P -positivo.

Silicio tipo P. Combinado con boro, al silicio le faltará un electrón en su última órbita para ser estable. A este electrón que hace falta se le llama hueco. Es posible que un electrón de un átomo cercano caiga en ese hueco.

Silicio tipo N. Con fósforo, el silicio tendrá un electrón extra. Este electrón se puede mover a través de la superficie del cristal con mucha facilidad.

Cuando el silicio se combina para formar material tipo N o tipo P podrá ser conductor si existe una fuerza contraria que

provoque un flujo de electrones. Aquí es necesario recordar la regla que dice que cargas iguales se repelen y cargas diferentes se atraen.

3.2 Interruptores y relevadores

Los interruptores de tipo mecánico —dispositivos como los transistores, al trabajar como dispositivos de conmutación son interruptores electrónicos— permiten o interrumpen el flujo de la corriente. El más sencillo es el SPST (*single pole, single throw*), esto es interruptor de un polo y un tiro. Su símbolo se muestra en la figura 3.2.1 (a). Hay interruptores de más contactos que también se muestran en la misma figura. Hay muchos otros tipos de interruptores, pero su uso es el mismo: interrumpir o permitir el flujo de corriente.

Un relevador es también un interruptor. Se le llama también interruptor electromagnético. Está formado por una bobina y un interruptor. Ver la figura 3.2.1 (b). Cuando una pequeña corriente fluye a través de la bobina del relevador se crea un campo magnético que hará que el interruptor se abra si es un interruptor NC —normalmente cerrado— o se cierre si el interruptor es un interruptor NO —normalmente abierto. La configuración de los contactos puede ser muy variada.

3.3 La resistencia, el capacitor y la bobina

Resistencias

Las resistencias se utilizan para limitar la corriente en un circuito. Su símbolo se muestra en la figura 3.1.1 (c). El valor de una resistencia está dado en ohms. Para reconocer su valor una

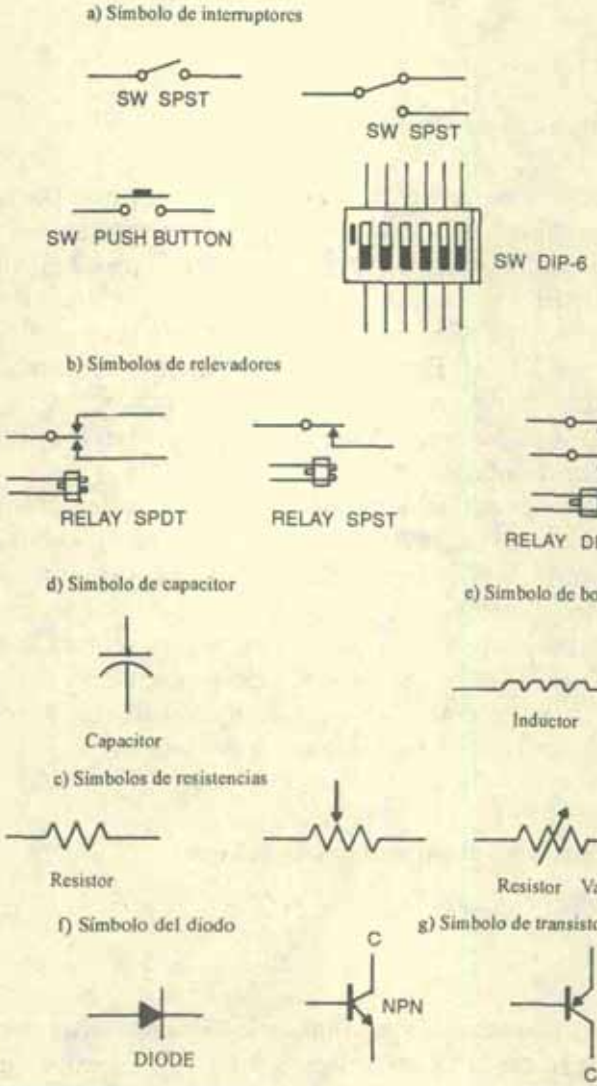


Figura 3.1.1. Símbolos de componentes electrónicos

resistencia tiene bandas de color en su cuerpo que indican su valor. El color de las dos primeras bandas indica el valor de la resistencia, que se va a multiplicar por la potencia que indique la última banda. Por ejemplo, si tenemos una resistencia cuyos colores son rojo, naranja y verde, su valor será 23 x 100,000 ohms, o 2,300 Kohms. El código de colores se muestra a continuación:

Color	1a. banda	2a.banda	3a. banda (multiplicador)
Negro	0	0	1
Café	1	1	10
Rojo	2	2	100
Anaranjado	3	3	1,000
Amarillo	4	4	10,000
Verde	5	5	100,000
Azul	6	6	1,000,000
Violeta	7	7	10,000,000
Gris	8	8	100,000,000
Blanco	9	9	

PRINCIPALES TIPOS DE RESISTENCIAS

- Carbón
- Alambre
- Película metálica

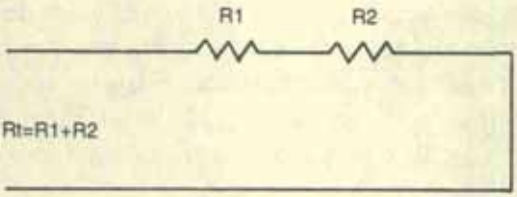
Las resistencias se pueden conectar en serie o en paralelo. Si se conectan dos en serie su resistencia total será:

$$Rt = R1 + R2$$

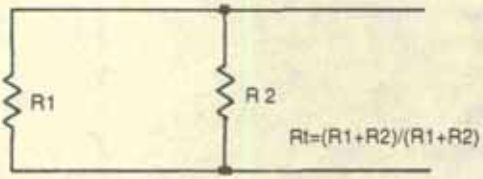
En cambio, si se conectan en paralelo su resistencia total será:

$$Rt = (R1 \times R2) / (R1 + R2)$$

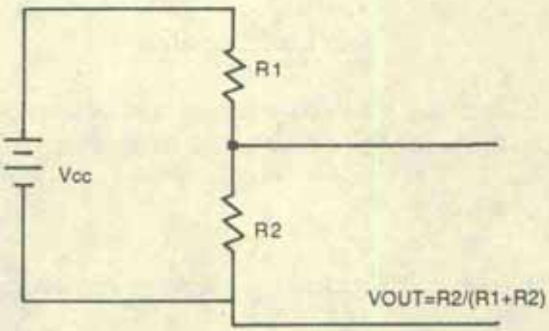
La figura 3.3.1 muestra cómo se pueden conectar las resistencias.



(A) CIRCUITO EN SERIE



(B) CIRCUITO EN PARALELO



(C) VOUT DIVISOR DE TENSION

Figura 3.3.1. Circuitos resistivos conectados en serie, en paralelo y divisor de tensión

El divisor de voltaje es un circuito muy importante. El voltaje de salida se determina de la siguiente manera:

$$Vs = Vin\,(R2 / (R1 + R2)),$$
$$Vin = IR1 + IR2,\ Vin - Vs = IR1,\ Vs = Vin - IR1,$$
$$I = Vin / (R1 + R2),\ Vs = Vin - Vin\,(R1 / (R1 + R2)),$$
$$Vs = Vin\,(R1 + R2 - R1) / (R1 + R2),$$

finalmente,

$$Vs = Vin\,(R2/(R1 + R2)).$$

Capacitores

Hay muchas clases de capacitores, pero todos hacen lo mismo, almacenar electrones. Están formados por dos conductores y un aislante en medio de los conductores. El aislante puede ser papel, plástico, mica, vidrio, cerámica, aire, etcétera. El conductor puede ser aluminio u otro metal. Si se aplica una tensión de un voltaje determinado en una de las terminales de un capacitor, éste se va a cargar. Si se quiere cargar con lentitud se puede conectar una resistencia en serie con el capacitor para que se limite la corriente de carga. El capacitor se descargará si se unen sus terminales. El capacitor se descargará más lentamente si se le conecta en serie una resistencia.

PRINCIPALES TIPOS DE CAPACITORES

- Electrolíticos
- Mica
- Cerámica
- Mylar
- Tantalio

La máxima carga del capacitor se especifica en faradios, lo que indicará el rango de voltaje que puede aceptar. Hay que asegurarse de que el capacitor que se piense utilizar exceda el rango de voltaje requerido en el circuito.

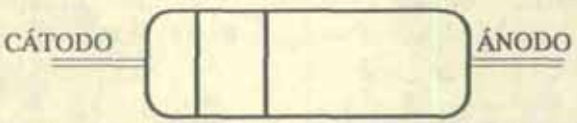
Bobina

Una bobina deja pasar libremente la corriente de DC. Se crea en ella entonces un campo magnético. Con bobinas se construyen motores, relevadores, transformadores, etcétera.

3.4 El diodo

El diodo es un dispositivo que sólo conduce energía eléctrica en una dirección. Para conducirla debe estar polarizado directamente con el ánodo a Vcc y el cátodo a tierra. Físicamente el diodo trae dibujada una banda que indica su cátodo. La figura 3.4.1 muestra un LED, que es un diodo emisor de luz, conectado directamente.

POLARIZACIÓN DE UN DIODO



Polarización directa: ánodo conectado al positivo de la fuente de DC y cátodo conectado a tierra o negativo de la fuente de DC. *El diodo permite el paso de la corriente eléctrica.*

Polarización inversa: ánodo conectado a tierra o negativo de la fuente de DC y cátodo conectado al positivo de la fuente de DC. *El diodo no permite el paso de la corriente eléctrica.*

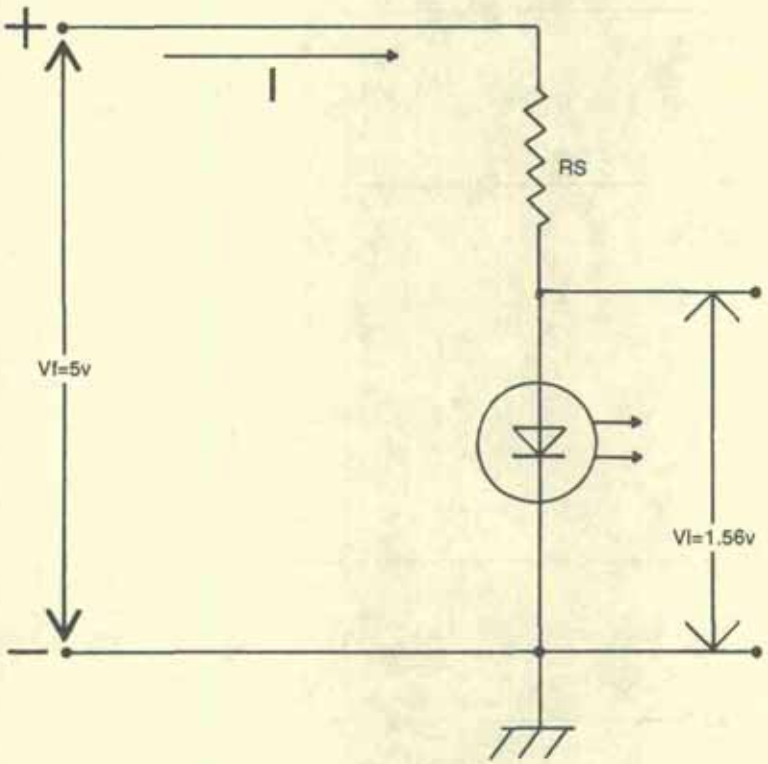


Figura 3.4.1. Diodo polarizado directamente

Ahora que ya se ha recordado cómo funcionan estos dispositivos, se podrán diseñar una compuerta OR y una compuerta AND con diodos. Cuando se diseña una compuerta con diodos, por lo regular se requiere un amplificador a la salida para poder mantener el nivel lógico correcto, puesto que una porción de la señal de voltaje se pierde en el diodo. La compuerta OR con diodos se muestra en la figura 3.4.2.

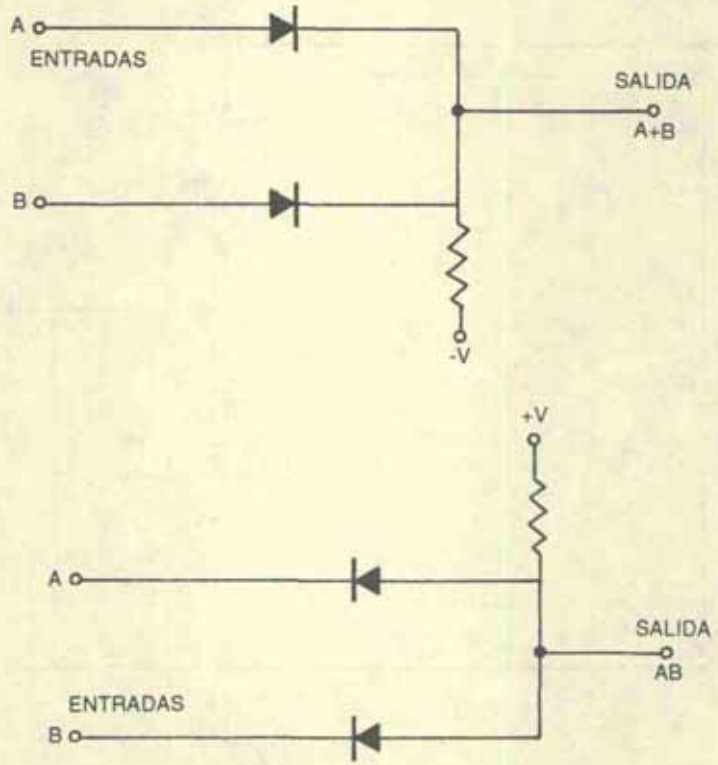


Figura 3.4.2. Compuertas AND y OR con diodos.

La figura 3.4.2 (a) muestra un circuito OR. Si una señal lógica con valor de 1 se aplicara a cualquiera de las entradas A o B, un 1 lógico aparecerá en la salida. Una señal positiva de voltaje pasará a través del diodo, que estará polarizado directamente. Habrá una caída de voltaje en la resistencia y una señal positiva en la salida. El voltaje de salida será de 0.6 volts menos que el

voltaje de entrada, debido a que en el diodo habrá una pequeña caída de voltaje.

La figura 3.4.2 (b) muestra un circuito AND con diodos. Cuando una señal positiva está aplicada en ambas entradas A y B, una señal lógica de 1 aparece a la salida. Cuando se aplica una señal lógica de 0 a las entradas A y B, habrá un voltaje de caída en la resistencia de carga, y la salida será un 0 lógico. Si una señal lógica de 1 es aplicada a cualquiera de las entradas, en ese diodo en particular no habrá flujo de corriente pero, puesto que el flujo de corriente no está bloqueado por el otro diodo, la salida se mantendrá en 0.

3.5 El transistor bipolar

Los transistores bipolares pueden ser npn o pnp, como se muestra en la figura 3.5.1. Se construyen con material semiconductor de germanio o de silicio. Los transistores de los circuitos integrados están hechos de silicio y son npn. C es el colector del transistor, B es la base y E el emisor.

La figura 3.5.1 muestra las características típicas de un transistor de silicio npn de emisor común. El circuito es un inversor simple con dos resistencias y un transistor. La corriente marcada I_C fluye a través de la resistencia R3 y el colector del transistor. La corriente I_B fluye a través de la resistencia R2 y la base del transistor. El emisor está conectado a tierra y su corriente I_E es igual a $I_C + I_B$.

El suministro de voltaje está entre Vcc y tierra. La entrada está entre el punto A y tierra; la salida entre el punto A' y tierra. Se supone una dirección positiva de las corrientes como se indica. Así fluye en forma normal la corriente en un transistor npn. Las corrientes de colector y de base son positivas cuando fluyen dentro del transistor. La corriente de emisor es positiva cuando fluye fuera del transistor.

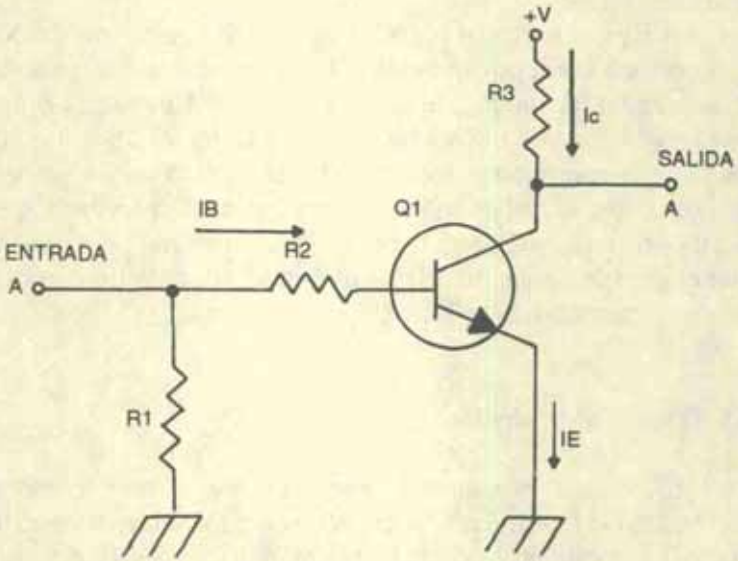


Figura 3.5.1. El transistor conectado como inversor

V_{CE} = Voltaje entre el colector y el emisor.
V_{BE} = Voltaje entre la base y el emisor.

El transistor puede estar en corte, región activa y saturación. A continuación se explican cada uno de ellos.

La unión base-emisor está polarizada directamente cuando el voltaje base-emisor es positivo, y está polarizado inversamente cuando el voltaje base-emisor es negativo. Si el voltaje base-emisor es menor que 0.6 volts se dice que el transistor está en corte y no fluye corriente de base. Cuando la junta base-emisor está polarizada directamente con un voltaje mayor de 0.6 volts, el transistor conduce y la corriente de base comienza a elevarse

con gran rapidez, en tanto que el voltaje base-emisor cambia muy poco.

El voltaje base-emisor a través de un transistor rara vez excede de 0.8 volts. Cuando el voltaje entre la base y el emisor es menor que 0.6 volts, el transistor está en corte con una corriente de base igual a 0 y fluye una corriente despreciable en el colector. El circuito de colector a emisor se comporta entonces como un circuito abierto. En la región activa el voltaje entre el colector y el emisor puede tener cualquier valor entre 0.8 volts y Vcc. La corriente del colector puede calcularse siendo aproximadamente igual a la corriente de base por la ganancia del transistor. La ganancia es un parámetro del transistor cuyo símbolo es h_{fe}. La máxima corriente del colector no depende de la corriente de base sino del circuito externo conectado al colector. Esto se debe a que el voltaje entre el colector y el emisor siempre es positivo y su valor más bajo posible es 0. Por ejemplo, en el inversor mostrado en la figura 3.5.1 la máxima corriente en el colector se obtiene al hacer el voltaje entre el colector y el emisor igual a 0, de tal manera que la corriente de colector es igual a:

$$Ic = Vcc / R3$$

Cuando $h_{fe}\ I_B$ es mayor que la corriente de colector, se dice que el transistor está en la región de saturación. Idealmente el voltaje entre el colector y el emisor, cuando el transistor está en saturación, debería ser 0, pero en forma normal es cerca de 0.2 volts. Los parámetros típicos de un transistor npn son en la región:

De corte. El voltaje entre la base y el emisor es menor de 0.6 volts, el voltaje entre colector-emisor es un circuito abierto, y la corriente de base y la corriente de emisor son iguales a 0.

Activa. El voltaje entre la base y el emisor es de 0.6 a 0.7 volts, El voltaje entre el colector y el emisor es mayor de 0.8 volts, y

la corriente de colector es igual a la corriente de base multiplicada por la ganancia.

De saturación. El voltaje entre la base y el emisor es de 0.7 a 0.8 volts, el voltaje entre el colector y emisor es de 0.2 volts y la corriente de colector es menor que h_{fe} I_B.

Para demostrarlo con un ejemplo considérese que los parámetros del circuito que aparecen en la figura 3.5.1 son los siguientes: R3 = 1K, R2 = 22K, Vcc = 5V, h_{fe} =50, Vin (max) = 5V y Vin (min) = 0.2V. Si Vin = 0.2V, entonces el V_{BE} es menor de 0.6 V; por lo tanto, el transistor está en corte. El circuito colector-emisor se comporta como un circuito abierto, de modo que el voltaje de salida Vout=5V. Si Vin= 5V, el voltaje entre la base y el emisor es mayor de 0.6 volts. Supóngase que es exactamente igual a 0.7 voltios. Se calcula entonces la corriente de base:

$$I_B = (Vin - V_{BE}) / R2 = (5 - 0.7) / 22K = 0.195 \text{ mA}$$

I_{CS} es la corriente de colector en saturación, o la corriente máxima de colector. Si el voltaje entre el colector y el emisor es igual a 0.2 volts, entonces:

$$I_{CS} = (Vcc-Vce) / R3 = (5 - 0.2) / 1 \text{ K} = 4.8 \text{ mA}$$

Se puede ahora verificar que el transistor está en saturación porque:

$$I_B h_{fe} = 0.195 \times 50 = 9.75 \text{ mA} > 4.8\text{mA} = I_{CS}$$

Esto demuestra que el circuito se comporta como un inversor porque cuando el Vin es 5 volts, el Vout es igual a 0.2 Volts.

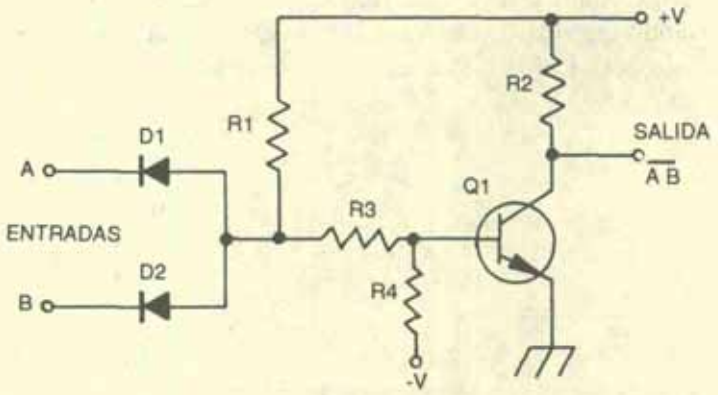


Figura 3.5.2. Compuerta NAND utilizando diodos y un transistor

Se pueden hacer compuertas combinando diodos y transistores, por ejemplo la figura 3.5.2 muestra una compuerta NAND compuesta por una compuerta AND hecha de diodos y seguida por un inversor. Aunque sólo se muestran dos entradas, es posible agregarle más sin degradar el rendimiento del circuito. El transistor invierte y amplifica la señal. Cuando se aplica un 0 a las entradas A y B, la corriente fluirá de +V a tierra pasando a través de los diodos. La base del transistor Q1 estará polarizada negativamente a través de R4 y el transistor estará en corte. Entonces la salida del circuito será casi igual a Vcc. Si sólo se aplica un 1 a la entrada, mientras la otra entrada está en 0, la corriente de la fuente de voltaje continuará fluyendo a través de R1 y a través del diodo que está conectado a 0. El voltaje en la unión de R1 y R3 será tan pequeño que no afectará al transistor Q1. Este transistor permanecerá en corte, y la salida del circuito seguirá siendo igual a 1.

Si aplicamos una señal lógica de 1 en ambas entradas no permitiremos el flujo de corriente a través de los diodos, la

corriente fluirá a través de R1, R3 y el emisor del transistor, causando que Q1 conduzca. Cuando Q1 conduce la salida del circuito es 0. Por lo tanto, este es un circuito NAND.

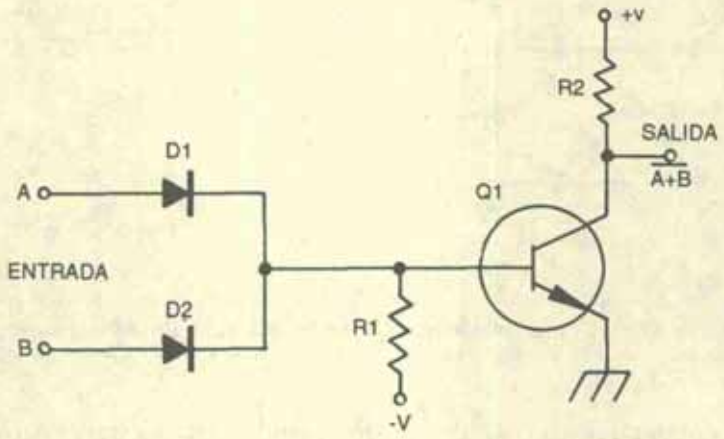


Figura 3.5.3. Compuerta NOR que utiliza diodos. Un 1 lógico en cualquiera de las entradas producirá un 0 lógico en la salida.

La figura 3.5.3 muestra una compuerta NOR, también formada por dos diodos y un transistor que invierte la salida.

Una señal lógica de 1 en cualesquiera de las entradas hará que el transistor esté en saturación y conduzca. Si Q1 conduce, la salida estará casi a tierra creando una señal lógica de 0. Si ambas entradas son 0, R1 causará que la base de Q1 también esté en 0. Por lo tanto, el transistor está en corte y no conduce. Cuando esto ocurre el transistor está en corte y una señal lógica de 1 estará presente a la salida del circuito. Por lo tanto, este es un circuito NOR.

Hay una familia de circuitos integrados que ya rara vez se utiliza, cuya configuración interna se realiza con diodos y tran-

sistores. A esta familia se le llama DTL, que significa Diode-Transistor Logic, esto es, lógica de diodo-transistor.

Hasta ahora se han visto implementaciones de compuertas con dispositivos como transistores, diodos y resistencias que se interconectan mediante alambres y circuitos impresos. La ventaja de utilizar circuitos integrados es que se forman sobre un solo dado de silicio. Por ello son más económicos, utilizan menos espacio, y consumen menos energía.

La complejidad de un circuito en un dado actualmente es increíble. La forma de fabricar un IC es muy similar a la de un transistor. Por lo tanto, el costo es casi igual, además de que se eliminan los costos de interconexión y las resistencias y otros componentes discretos. Las familias más importantes son DTL, RTL, TTL, ECL y CMOS. A continuación se analizará cómo funcionan estas familias de ICs.

La figura 3.5.4 muestra una compuerta DTL. El circuito es muy parecido al de compuerta NAND de la figura 3.5.2. El transistor adicional hace que el circuito tenga una ganancia mayor.

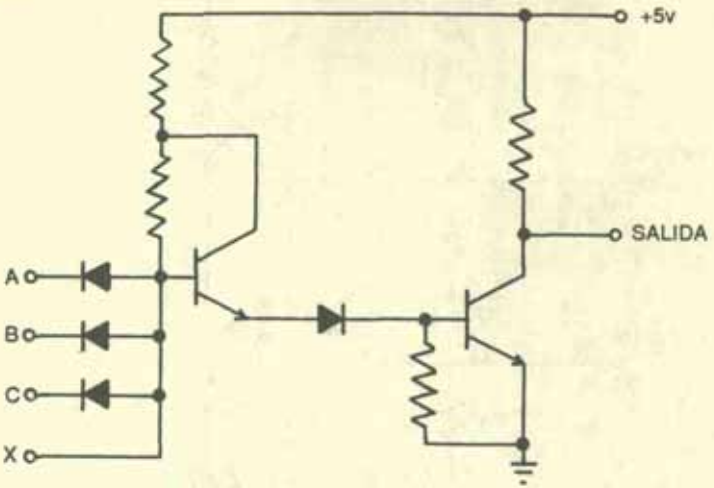


Figura 3.5.4. Compuerta NAND con lógica DTL.

Trabaja a 5 volts. El número de entradas puede ser entre dos y diez. De ello depende que un IC pueda traer de una a cuatro compuertas. La entrada X es una entrada que permite que se conecten diodos adicionales para incrementar el fan-in o expandir el número de entradas a 20. El fan-out o número de entradas similares que puede alimentar la salida es de ocho a diez. El tiempo que tarda la compuerta en pasar de nivel 0 a nivel 1 es de 30 nseg y la disipación típica de potencia por compuerta es de 10 mW por compuerta. Otras compuertas que se pueden obtener son AND, OR y NOR.

Compuertas con transistores

Compuertas NAND. Cuando se diseñan compuertas con transistores, por lo regular el transistor invierte la salida. Si se conectan dos transistores como se muestra en la figura 3.5.5, se tendrá una compuerta NAND.

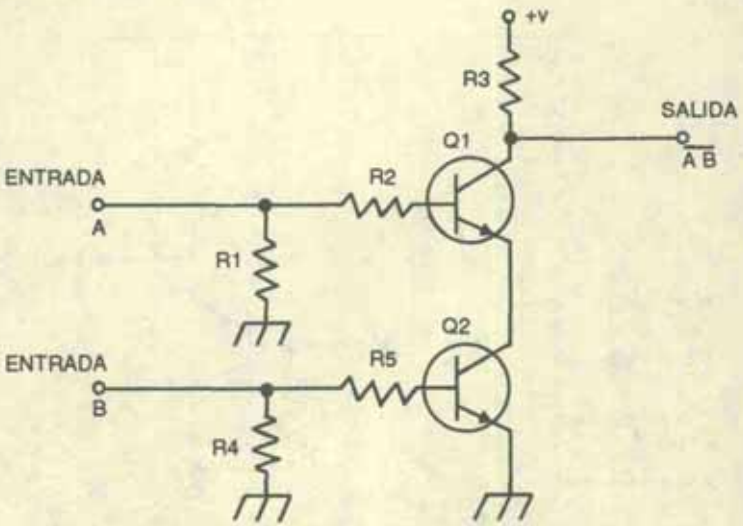


Figura 3.5.5. Circuito básico NAND. Conexión en serie.

Cuando los transistores están conectados en serie se deberá tener un 1 lógico en ambas entradas para obtener un 0 a la salida. Por lo tanto este circuito es una compuerta NAND. Cuando A y B están en 0, tanto Q1 como Q2 estarán en corte. La salida del circuito alcanzará el valor de la fuente a través de la resistencia R3, dándonos una salida lógica de 1. Si un 1 lógico es aplicado en la entrada A, mientras que la entrada B se mantiene en 0, Q1 conducirá pero Q2 permanecerá en corte. Por tanto, la salida se mantendrá en 1.

El circuito que se muestra en la figura 3.5.6 es muy parecido al que se muestra en la figura 3.5.5, excepto porque en éste se utilizan pocos componentes. Este es un circuito DCTL (*direct-coupled-transistor logic*) que significa lógica de transistores acoplados directamente. Este tipo de ICs fue muy popular porque utilizaba muy pocos componentes y reducía el costo de manufactura.

Aquí los transistores operan en la región de saturación. Para que haya estabilidad y flexibilidad, el voltaje entre el colector y el emisor deberá ser menor que el voltaje entre la base y el emisor. Estos arreglos dan como resultado altas velocidades y bajos suministros de energía. Son circuitos muy eficientes.

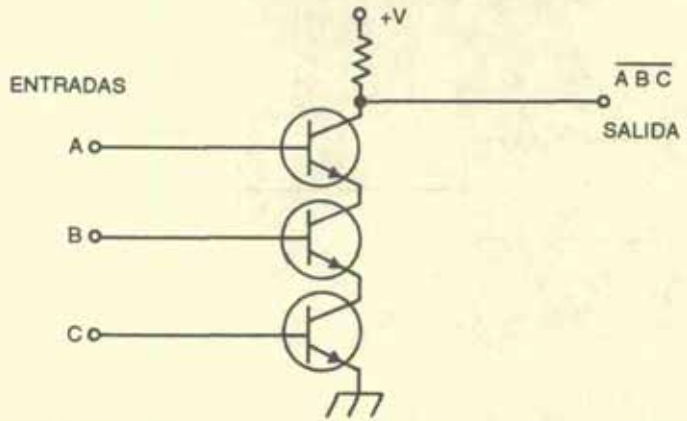


Figura 3-5.6 Compuerta NAND DCTL

Como en el circuito anterior, los transistores están conectados en serie, por lo tanto tendrá que haber un 1 en A, B y C para que la salida cambie a 0. El circuito es una compuerta NAND.

Compuertas NOR.- Un circuito NOR es el que produce la salida de un circuito OR invertido. Como ya se mencionó antes, cuando se diseñan compuertas con transistores la salida se invierte automáticamente. La figura 3.5.7 muestra una compuerta NOR básica.

Cuando se recibe un 1 lógico en cualquiera de las dos entradas A o B, o en ambas, el transistor se polariza directamente y conduce. Por lo tanto la salida será un 0 lógico. Si un 0 lógico está presente en ambas entradas el transistor estará en corte, el voltaje colector-emisor será casi igual al de la fuente. Se tiene entonces un 1 lógico a la salida. Este circuito nos da una salida OR invertida. Las resistencias forman la compuerta OR, mientras que el transistor amplifica e invierte la señal. A estos circuitos se les conoce como circuitos RTL (*resistor-transistor logic*), que significa lógica de resistencia y transistor. Esta también es una familia de ICs que ya no es muy usada.

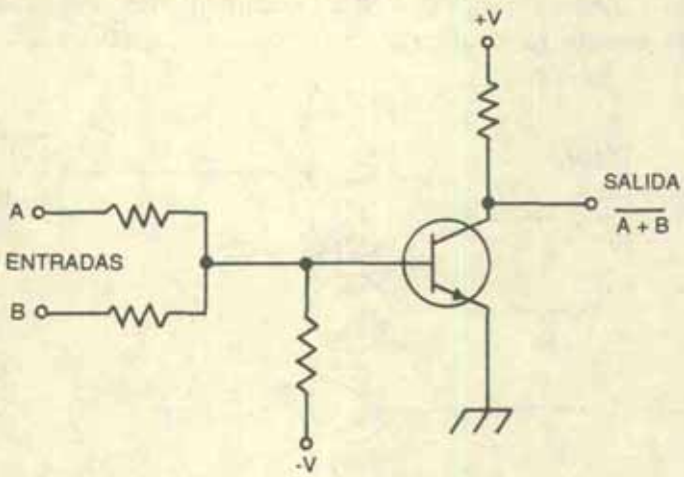


Figura 3.5.7. Compuerta básica NOR

Si se conecta en paralelo con la resistencia un capacitor, es posible que se aumente la velocidad de conmutación. La figura 3.5.8 muestra uno de estos circuitos, comúnmente conocido como un circuito RCTL (*resistor-capacitor-transistor logic*): lógica de resistencia-capacitor-transistor.

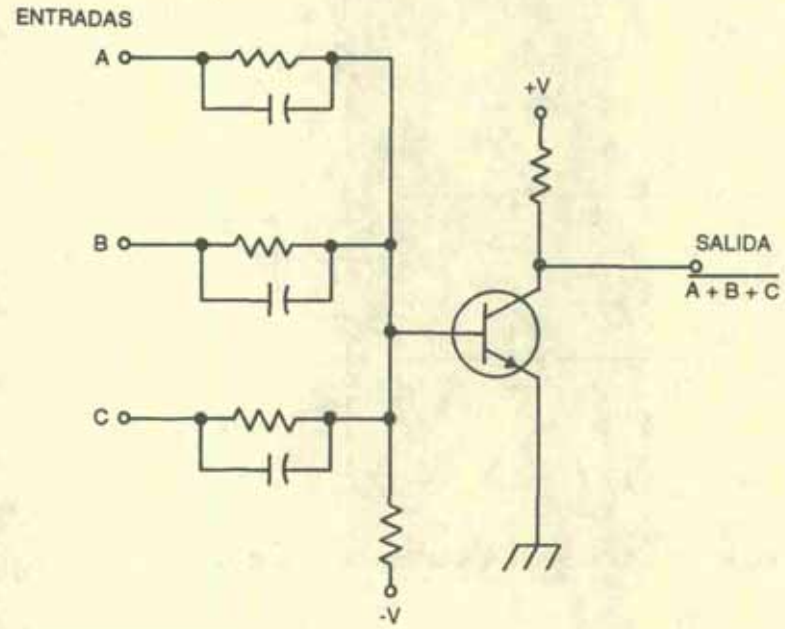


Figura 3.5.8. Compuerta NOR con lógica RCTL

El circuito de la figura 3.5.8 es muy similar al de la compuerta RTL de la figura 3.5.7. Los capacitores suministrarán carga a la base. Esto incrementa la corriente en la base cuando se recibe una señal lógica, y esto da como resultado velocidades rápidas de conmutación en los transistores. Si la velocidad no es muy importante se omiten los transistores para que el circuito resulte más económico.

La figura 3.5.9 muestra una compuerta NOR la cual utiliza un transistor para cada entrada.

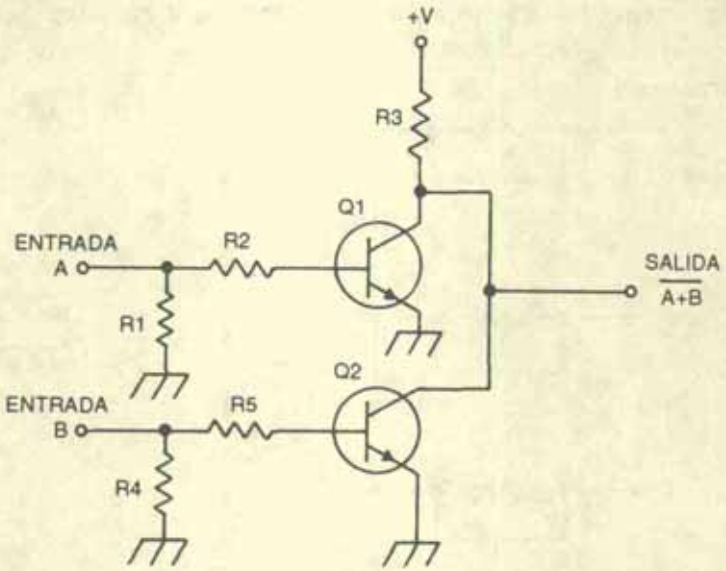


Figura 3.5.9. Compuerta NOR utilizando transistores npn.

El circuito utiliza dos transistores para la función NOR. En la figura se muestran dos entradas, aunque es posible agregar más entradas siempre y cuando los colectores de los transistores estén directamente acoplados en un mismo punto.

Cuando haya una señal lógica de 1 en cualquiera de las dos entradas A o B, el transistor asociado con esa entrada conducirá, teniendo a la salida un 0 lógico. Otra vez la señal es invertida; el circuito es una compuerta NOR. Cuando las entradas están a tierra, los transistores no conducen. Por lo tanto, a la salida se tendrá un 1 lógico.

Otra manera de construir una compuerta NOR con un mínimo de componentes, es la que se muestra en la figura 3.5.10.

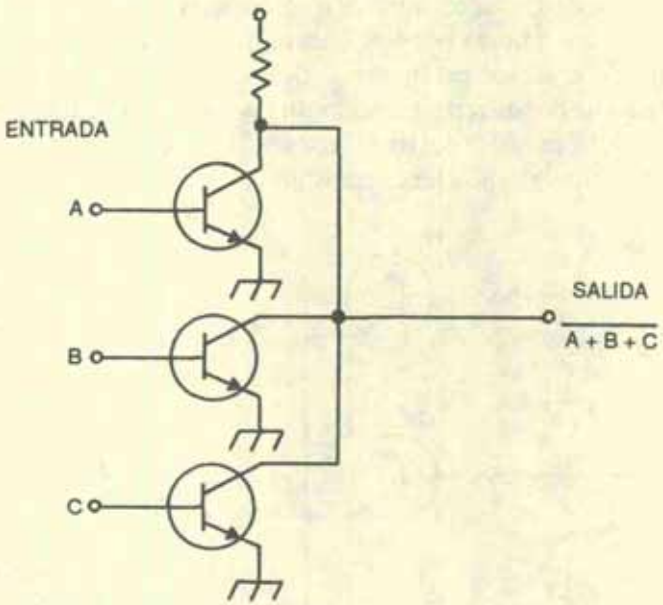


Figura 3.5.10. Circuito de una compuerta NOR utilizando los menos componentes posibles.

Este circuito también se parece a las anteriores compuertas NOR antes descritas. Es un circuito DCTL, en el que los transistores operan en saturación. Unicamente cuando las tres entradas sean igual a 0 la salida será igual a 1.

Compuertas AND.- Cuando se diseñan compuertas con transistores, con frecuencia se requiere que la salida de la compuerta se invierta. La figura 3.5.11 muestra una compuerta NAND acoplada a un circuito inversor, formando un circuito AND.

Los transistores Q1, Q2, y Q3 están polarizados para permitir que la corriente fluya cuando haya un 0 en las entradas. Además se conectan las bases de los transistores a través de una resistencia a tierra, y todos los colectores de los transistores PNP están unidos y conectados a tierra por R9. Puesto que el voltaje de base y el voltaje de colector están casi a tierra, esto hará que los tres transistores conduzcan. Cuando un 0 se aplica a cualquiera de las entradas, la salida del transistor será 1. Esto hará que Q4 esté en corte y en la salida tendremos un 0.

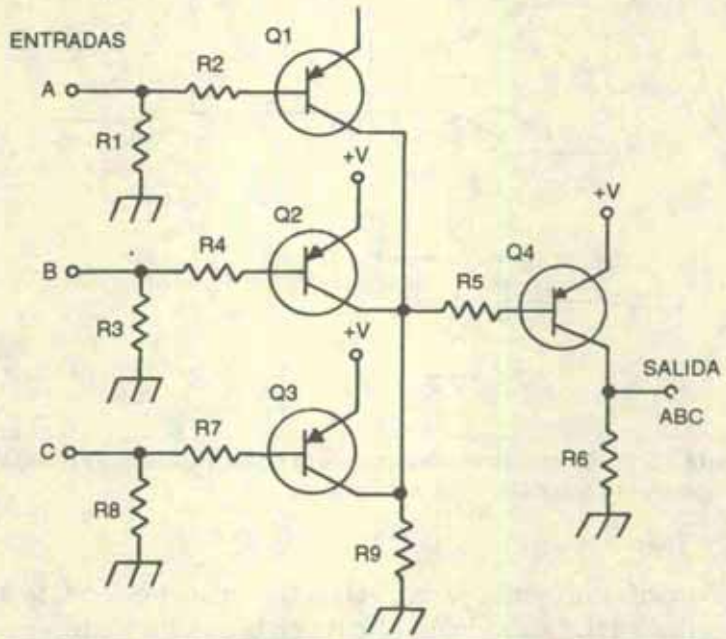


Figura 3.5.11. Compuerta AND utilizando transistores PNP

Supóngase que aplicamos un 1 lógico a la entrada A. Una señal positiva aplicada a la base de Q1, hará que el transistor esté en corte. Pero como en Q2 y en Q3 todavía tenemos un 0, estos dos

transistores están conduciendo. Por lo tanto, no habrá cambio aparente en la base del transistor Q4, y la salida del circuito continuará en 0.

Esto significa que si nosotros queremos un 1 en la salida, se deberá aplicar un 1 en las entradas A, B, y C.

Compuertas OR. Una compuerta OR se muestra en la figura 3.5.12. Es muy similar a las compuertas NOR analizadas con anterioridad pero este circuito tiene agregado un inversor.

Si se aplica un 0 lógico en las entradas A, B, y C, los transistores Q1, Q2 y Q3 estarán en corte, y la salida de cada colector será casi igual al voltaje de la fuente. La polarización positiva en la base de Q4 hace que éste conduzca, haciendo que la salida sea igual a 0.

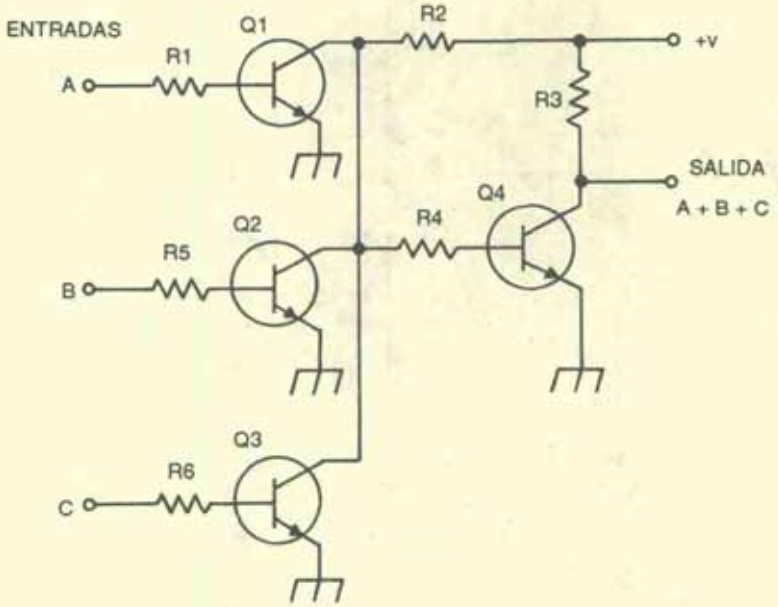


Figura 3.5.12. Compuerta OR de tres entradas.

Cuando una señal lógica de 1 es aplicada a la entrada A, el transistor uno comenzará a conducir. Como el colector de Q1 está aplicado a la base de Q4 y el colector de Q1 está casi a tierra, entonces Q4 está en corte. En la salida se tendrá un 1. Lo mismo pasa si se aplica un 1 a A o a C. En cuanto haya un 1 en cualquiera de las entradas se tendrá un 1 presente en las salidas. Es claro que el circuito representa una compuerta OR.

Combinación de compuertas AND-OR. La figura 3.5.13 muestra un circuito el cual combina dos compuertas OR y una AND. El circuito tiene cuatro entradas A, B, C, y D. Con un 0 en todas las entradas, ni Q1 ni Q2 conducirán. Esto producirá un 0 en la salida del circuito.

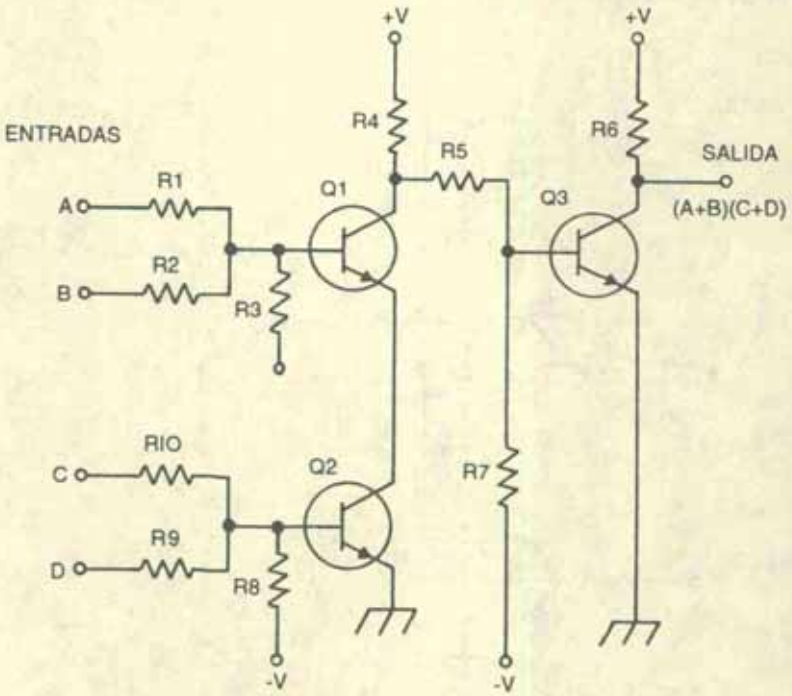


Figura 3.5.13. Circuito combinado AND-OR.

Si un 1 es aplicado en A o en B o en ambas, Q1 conducirá pero, puesto que Q2 está en corto, no habrá flujo de corriente. En Q3 no habrá cambio y la salida seguirá siendo 0. Esto significa que para obtener un 1 en la salida, se deberá tener un 1 en A o en B y un 1 en C o en D. Expresado en términos del álgebra de Boole será: (A + B) (C + D).

Se pueden conectar varios niveles de compuertas; sin embargo, se tiene que tomar en cuenta el consumo de energía. Por ejemplo, obsérvese la figura 3.5.14. Cuando el primer transistor está saturado, fluirá una corriente de carga. Si se conectan varias compuertas en paralelo, cada una contribuirá con una I_L similar.

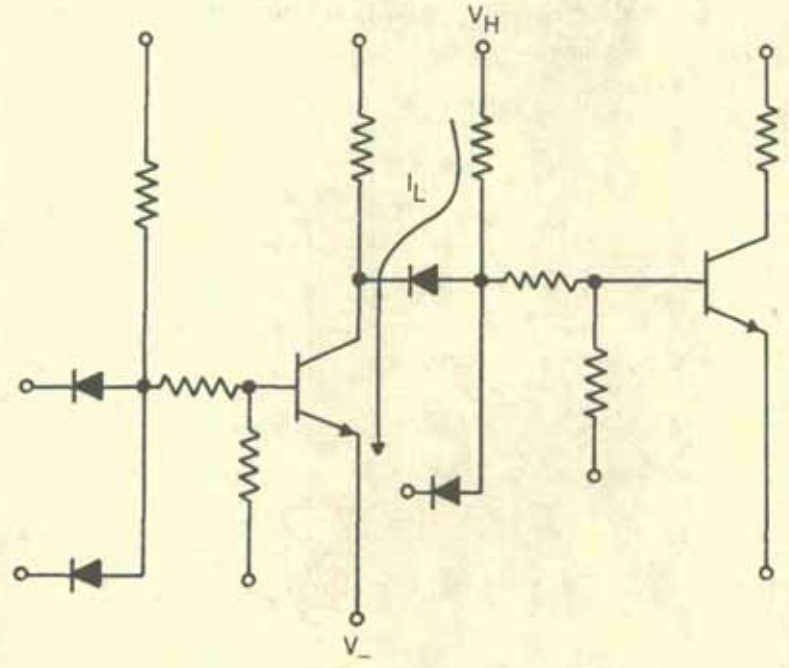


Figura 3.5.14. Carga de compuertas conectadas en paralelo

Velocidad y retardo. Es muy importante que los circuitos lógicos tengan una respuesta de conmutación corta. La figura 3.5.15 muestra un inversor con el emisor a tierra. Se ve cómo responde a un impulso positivo en la base, pasando de corte a saturación y luego de saturación a corte. Hay varios factores que determinan esta respuesta. Esto tiene que ver con el material con que esté hecho el circuito. Por ello, para conocer los voltajes de transición de las diferentes familias de ICs es necesario ver los manuales de sus fabricantes.

En la figura 3.5.15 se muestra un pulso donde:

t_d = *delay time*, que es el tiempo que tarda el transistor en detectar que hay una conmutación en la base de 0 a 1.

t_r = *rise time*, tiempo que tarda la salida en pasar de 0 a 1.

t_s = *set-up time*.

t_f = *fall time*.

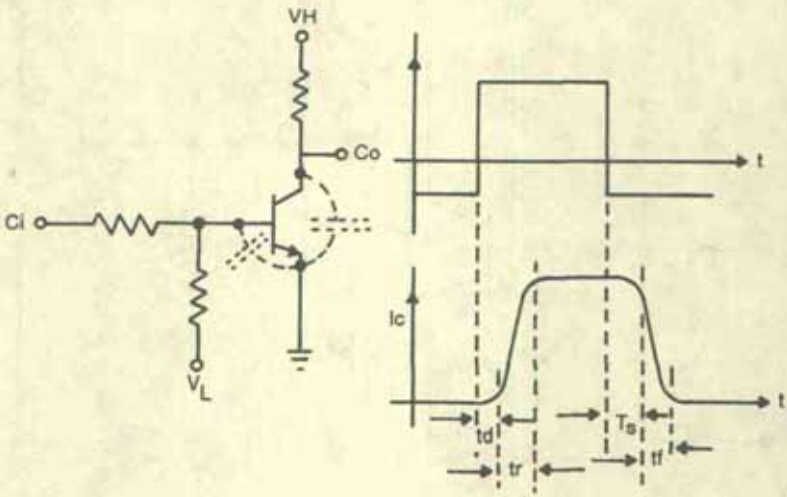


Figura 3.5.15. Tiempo de conmutación de un transistor conectado como inversor

3.6 El transistor de efecto de campo

Esta sección se referirá en particular al transistor MOSFET. Este es un transistor que se utiliza en el diseño de circuitos digitales. El transistor MOSFET (*metal-oxide semiconductor field-efect transistor*) físicamente ocupa menos espacio que los transistores bipolares. Su símbolo se muestra en la figura 3.6.1.

Figura 3.6.1. Símbolos de los transistores MOSFET

En el transistor MOSFET la G significa gatillo, la D significa drenaje, y la S significa fuente.

El transistor de canal P consta de una capa pesada tipo n, el drenaje y la fuente van conectados cada uno a un dopado con impurezas tipo p. La región n sirve como canal entre las dos regiones tipo p. La compuerta es una placa de metal separada del canal por un dieléctrico. Un voltaje negativo en la terminal de la compuerta causa un campo eléctrico inducido en el canal. Si aumenta el voltaje negativo en la compuerta, la corriente fluirá de la fuente al drenaje. Su impedancia de entrada es muy alta.

En el canal n la terminal del drenaje se conecta a un voltaje positivo. Cuando el voltaje en la compuerta es menor que 2 volts

no fluye corriente en el canal. Cuando el voltaje es mayor de 2 volts fluye una corriente positiva desde el drenaje a la fuente.

En el símbolo del MOSFET la dirección de la flecha indica la dirección del flujo de corriente. En el canal p es desde la fuente al drenaje, y en el canal n es desde el drenaje a la fuente.

Otra utilidad de los MOS es que se pueden construir resistencias con valores diferentes fijando diferentes longitudes y anchos de canal.

La figura 3.6.2 muestra tres circuitos lógicos utilizando MOSFETs. Los circuitos utilizan una fuente de 5 volts.

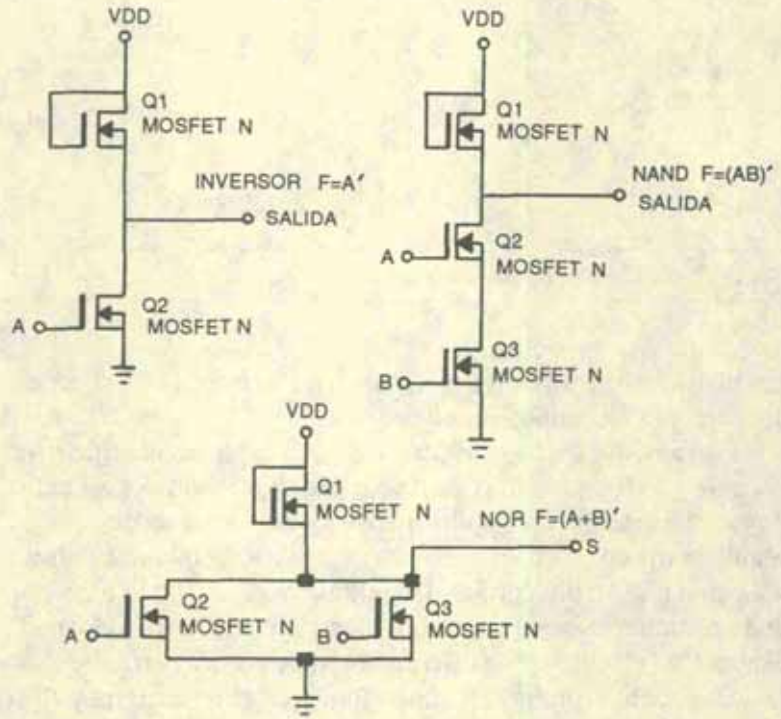


Figura 3.6.2. Circuitos lógicos con transistores MOSFET, canal N

La compuerta NAND que se muestra en la figura utiliza transistores en serie. Las entradas A y B deben ser 1, de manera que los dos transistores conduzcan y la salida sea 0. Si cualquier entrada es 0, el transistor correspondiente se pone en corte y la salida es 1. La resistencia en serie de los MOS que forman la compuerta deberá ser menor que la resistencia del MOSFET que sirve como resistencia de carga. La compuerta NOR que se muestra en la figura utiliza transistores en paralelo. Si cualquier entrada es alta el transistor conduce y la salida será igual a 0. Si todas las entradas son 0, la salida será 1.

3.7 Lógica de emisor acoplado

Las tres últimas secciones de este capítulo se refieren a la configuración interna de los circuitos integrados. Cuando se diseñan circuitos lógicos su realizador, de acuerdo con sus necesidades, puede escoger entre varios tipos de circuitos. Los más populares son los ECL y los TTL, entre los que se encuentra la familia estándar (74xx), la de baja potencia (74Lxx), la Schottky (74Sxx), la de alta velocidad (74Hxx), la de baja potencia y Schottky (74LSxx), la de super alta velocidad (74Fxx),la CMOS (74Cxx) que es igual a los CMOS, y finalmente los CMOS. Factores importantes que los diferencian son su velocidad de conmutación y su inmunidad al ruido que es la capacidad que tienen los circuitos para discriminar señales aleatorias en las líneas de entrada. Este tiende a incrementarse al aumentar la velocidad de conmutación. La escala de integración también es un factor importante, así como la disponibilidad de circuitos con funciones específicas como ALUs, memorias, etcétera.

A continuación se muestra una tabla con las características más importantes que distinguen a estas familias. Cabe hacer la advertencia de que estas características cambien en poco tiempo debido a los adelantos de la tecnología.

SERIE	FAN-IN	FAN-OUT	DISIPACIÓN DE POTENCIA	RETARDO	RUIDO	FRECUENCIA EN ENTRADAS
			mW	nseg		
ECL	5	20	Alta	1-2	reg	3 GHz
TTL LS	8	10	2 mW	9.5	reg	45 MHz
TTL L	8	10	1 mW	33	reg	3 MHz
TTL S	8	10	19 mW	3	reg	125 MHz
TTL	8	10	10 mW	10	reg	35 MHz
TTL H	8	10	22 mW	6	reg	50 MHz
TTL F	8		4 mW	2	poco	5 GHz

Tabla 3.7.1: Características básicas de ICs digitales

FAN-IN. Es el número de entradas que tiene una compuerta. Por ejemplo, el 7400 es una compuerta NAND de dos entradas. En este caso su FAN-IN es igual a dos.
FAN-OUT. Es el número de cargas estándar que la salida de un dispositivo puede manejar sin degradar su operación.
ICs ECL. Significa lógica de emisor acoplado. Aquí los transistores trabajan en la región activa. Como se pudo apreciar en la tabla, se trata de una familia que trabaja a muy alta velocidad. Tiene la desventaja de que consume mucha energía y además es poco inmune al ruido. La figura 3.7.1 muestra una compuerta ECL que tiene dos salidas: una salida es para compuerta OR y la otra es para compuerta NOR. Debido a las desventajas que presenta hace tiempo que dejó de ser popular.

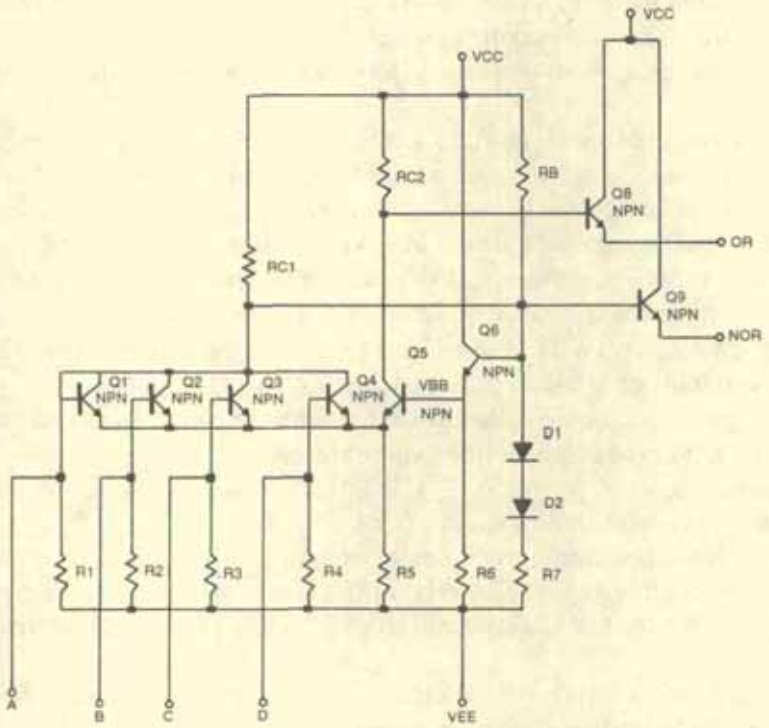


Figura 3.7.1. Lógica ECL

3.8 Lógica de transistor-transistor

Los circuitos integrados más populares son los TTLs (*transistor-transistor logic*), la serie 54xx y 74xx. Son casi idénticas, pero la serie 54xx trabaja de -55°C a 125°C y la serie 74xx trabaja de 0°C a 70°C. Son construidos por diferentes empresas,pero su funcionamiento es el mismo. Esta familia tiene la particularidad de utilizar transistores con emisores múltiples.

La siguiente es una lista de términos usados en los manuales:

Logica 1 voltaje de entrada Vin. Es el nivel de voltaje mínimo permitido. Cuando es aplicado a una de las entradas asegura que el dispositivo reconoce que es un nivel lógico de 1.
Logica 0 voltaje de entrada Vin. Es el máximo nivel de voltaje permitido. Cuando es aplicado a una de las entradas asegura que el dispositivo reconoce que hay un nivel lógico de 0.
Logica 1 voltaje de salida Vout. Es el mínimo nivel de voltaje permitido en la salida para indicar que hay un 1 lógico.
Logica 0 voltaje de salida Vout. Es el máximo voltaje de salida permitido para indicar que ésta representa un 0 lógico.
Frecuencia de operación. Es la máxima frecuencia con la que se garantiza que el dispositivo opere.

Hay tres configuraciones principales en los circuitos TTL dependiendo de su salida. A la configuración estándar se le llama poste-totem. Otra es la de colector abierto, la última es la de tres estados.
Poste-totem. Es la compuerta típica. Se muestra en la figura 3.8.1. Se divide en tres secciones principales: estado de entrada, de cambio de fase y de salida. El estado de entrada consiste en un transistor de emisor múltiple y una resistencia R1. La R1 determina la cantidad de corriente que fluirá hacia el emisor durante el tiempo que la entrada está en un nivel lógico de 0. El estado de cambio de fase consiste en un transistor Q2 y las resistencias R2 y R3. Este estado controla la condición de la salida, deter-

minando qué transistor (Q4 o Q3) deberá estar en saturación. Si Q2 está en corte, el transistor Q4 deberá estar en saturación y el transistor Q3 deberá estar en corte. Lo contrario sucederá si Q2 está en saturación. En todas las configuraciones este estado de cambio de fase es igual. Sin embargo los valores de las resistencias son elegidos en relación con la velocidad y el consumo de potencia.

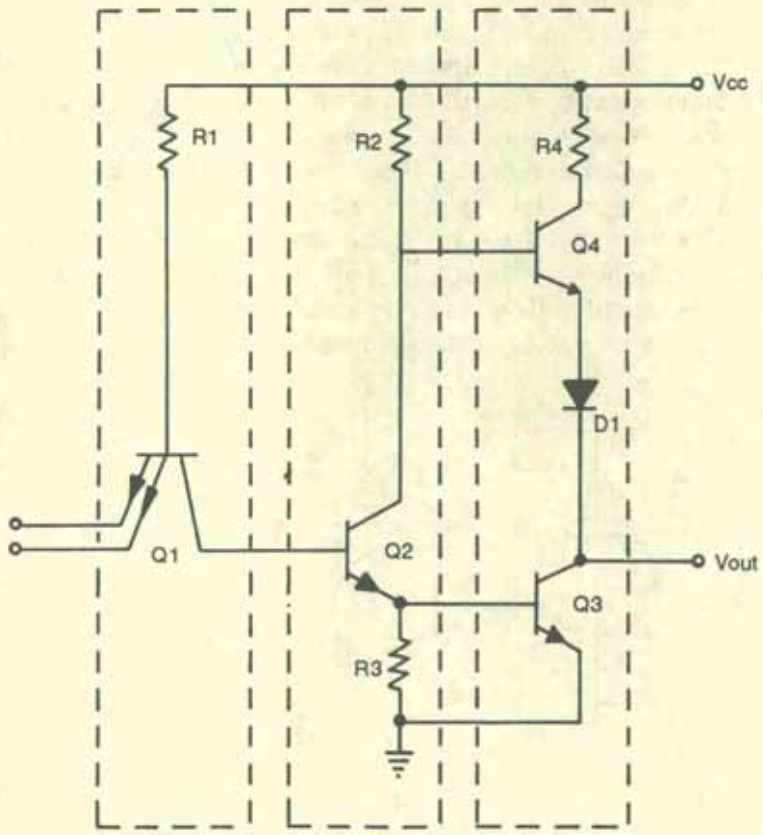


Figura 3.8.1. Compuerta TTL típica, configuración de TOTEM-POLE o poste-totem

El estado de salida consiste básicamente en la conexión de los transistores Q3 y Q4. Precisamente por la forma en que están acomodados estos dos transistores, a esta configuración se le llama de TOTEM-POLE o de poste-totem.

Colector abierto. Un dispositivo típico de colector abierto se muestra en la figura 3.8.2. Como se ve en la figura, en las salidas tienen el colector abierto. Una salida de una compuerta de colector abierto se puede conectar a otras para realizar lógica adicional, conectando una resistencia de la salida a Vcc, por lo regular de 2.2K. A este tipo de conexión se le llama de lógica alambrada. Si se conecta de esta manera un 7405, que es un IC con seis inversores con colector abierto, se obtendrá una compuerta NOR de seis entradas. Si se conectan cuatro compuertas NAND de colector abierto se obtendrá entonces la función de salida (A B + C D + E F + G H)'. La ventaja del colector abierto es que aumenta el FAN-OUT de 1 a 3. La desventaja es que es poco inmune al ruido y la velocidad disminuye. La figura 3.8.3 muestra un arreglo de lógica alambrada.

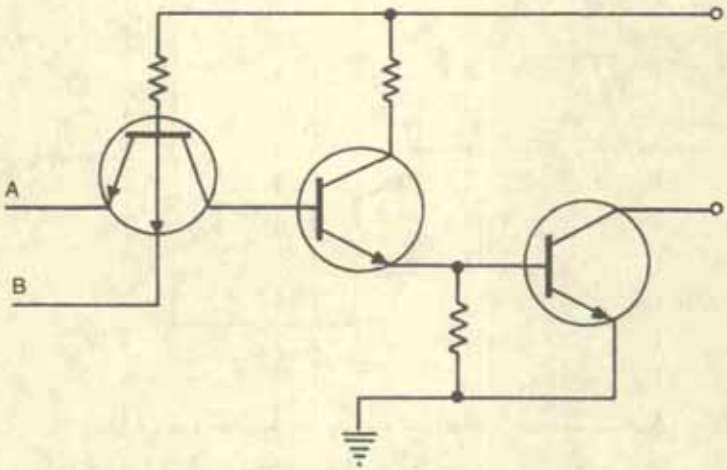


Figura 3.8.2. NAND colector abierto

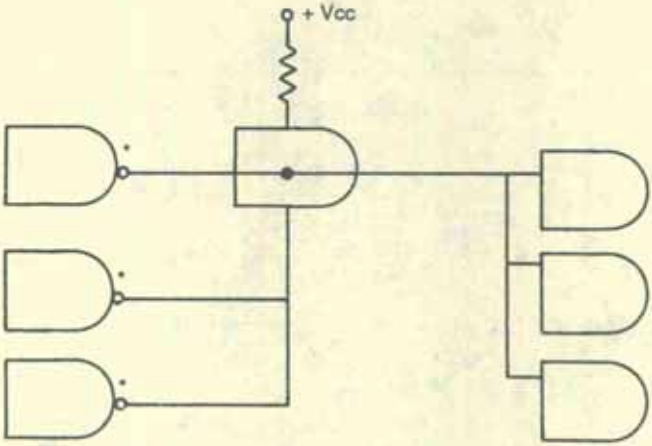


Figura 3.8.3. Lógica abierta alambrada

En colector abierto la salida será de 1 únicamente cuando todas las entradas son 0. De otra manera la salida será 0. Es importante hacer notar que sólo se pueden conectar todas las salidas juntas en colector abierto o en tres estados, y no en la familia TTL estándar.

Lógica de tres estados. La lógica de tres estados no tiene las desventajas del colector abierto. Esta lógica fue desarrollada por National Semiconductors. Es una herramienta muy poderosa para cualquier aplicación donde muchas compuertas lógicas tienen que intercambiar información, unas con otras en un sistema de bus común. Una aplicación obvia es la de las minicomputadoras, donde cualquier número de entradas y dispositivos de control deben ser capaces de "hablar" en forma selectiva entre ellas y en cualquier combinación de bus. La figura 3.8.4 muestra la configuración interna de esta lógica, que tiene una entrada de control. Cuando la entrada de control está activada

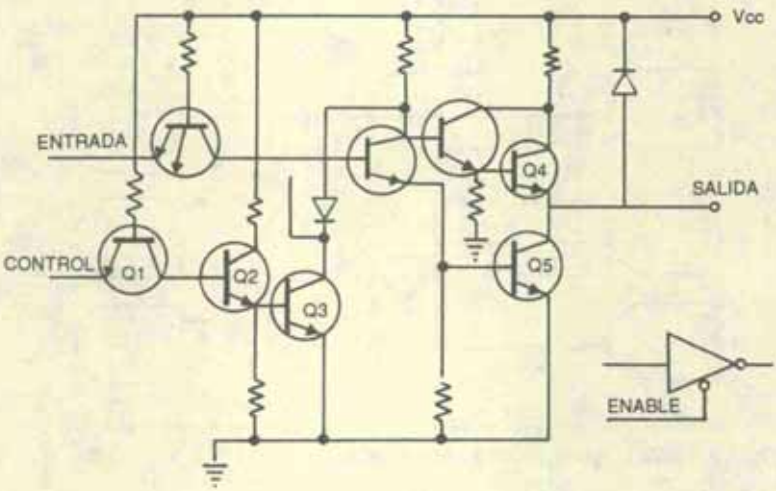


Figura 3.8.4. Configuración interna de la lógica de tres estados

la salida se comporta como una compuerta ordinaria TTL. Cuando la entrada de control se desactiva, el circuito interno está desconectado de la salida. La salida puede entonces tomar el valor que quiera y el circuito integrado es transparente a cualquier cosa que se conecte a la salida. La figura 3.8.5 muestra dos aplicaciones de la lógica de tres estados. En la figura 3.8.5 (a), se pueden usar las entradas de control como selectores de datos. La figura 3.8.5 (b) muestra un sistema orientado a bus, tal como se usa en una minicomputadora. Cualquiera de los dispositivos puede hablar con el otro en la misma línea. Por supuesto, sólo una línea puede comunicarse a la vez.

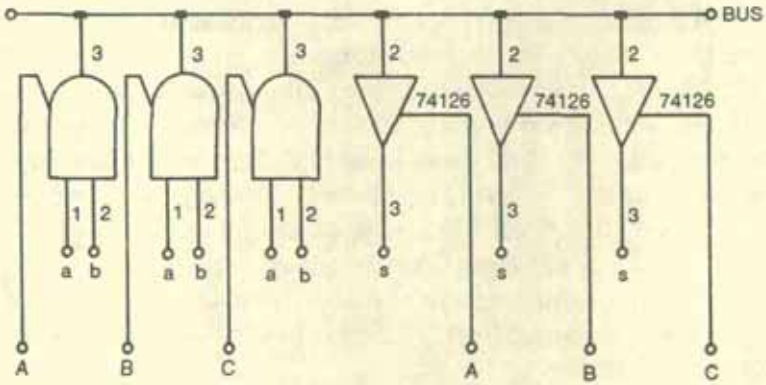


Figura 3.8.5 (a). Sistema de conexión simplificada

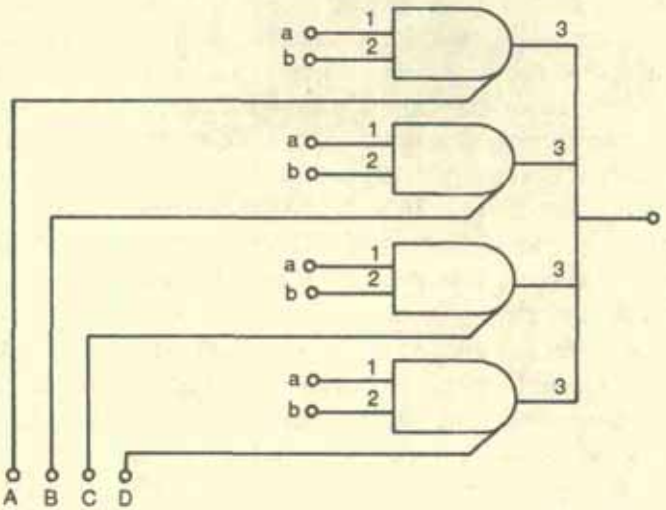


Figura 3.8.5 (b). Utilizando la lógica de tres estados

3.9 Lógica de semiconductores complementarios metal-óxido

La familia lógica más nueva es la CMOS (*Complementary-Metal-Oxide-Silicon*). CMOS tiene muchas ventajas importantes sobre las otras familias lógicas: económica, de bajo consumo de energía, entradas de circuito abierto, fan-out que no tiene límite, inmune al ruido y con gran variedad de diferentes dispositivos en SSI, MSI, LSI y VLSI. Cuando no hay conmutación la energía que se consume es casi 0. Son además adecuados para aprender circuitos digitales. La frecuencia de entrada máxima es de 5 Megahertz por segundo. Su desventaja es que sus salidas son muy sensitivas a las cargas externas, en particular a la capacitancia. Esto significa que hay que tener cuidado cuando se conectan CMOS en interfase.

La figura 3.9.1 (a) muestra un CMOS, o un inversor con MOS complementarios. Cuando la entrada es 0, el transistor de canal P conduce y el de canal N no. En este caso el canal P actúa como una resistencia de carga por cierto muy pequeña. Si se pone una carga a la salida no se consume corriente. Cuando la entrada es 1, el transistor de canal P no conduce y el transistor de canal N sí. La resistencia de la salida a tierra es muy pequeña. Otra vez no hay consumo de energía. Unicamente consume energía cuando hay un cambio de estado, porque se necesita cierta energía para cargar el capacitor que está en la compuerta.

Características de los CMOS. Todas las entradas a estos dispositivos son circuitos abiertos, muy fáciles de manejar. Sólo consumen corriente cuando van a conmutar. Por ello, la corriente de operación es muy baja, sobre todo en frecuencias bajas. Son también muy inmunes al ruido porque el cambio a la salida lo efectúan cuando la entrada es igual a 1/2 de Vcc. No necesitan una fuente super regulada y trabajan de +3 a +15 volts. Finalmente. sus salidas no crean ruido por sí mismas.

En la figura 3.9.1 se muestran las compuertas NAND y NOR de lógica CMOS. Para la compuerta NOR se tienen dos transistores

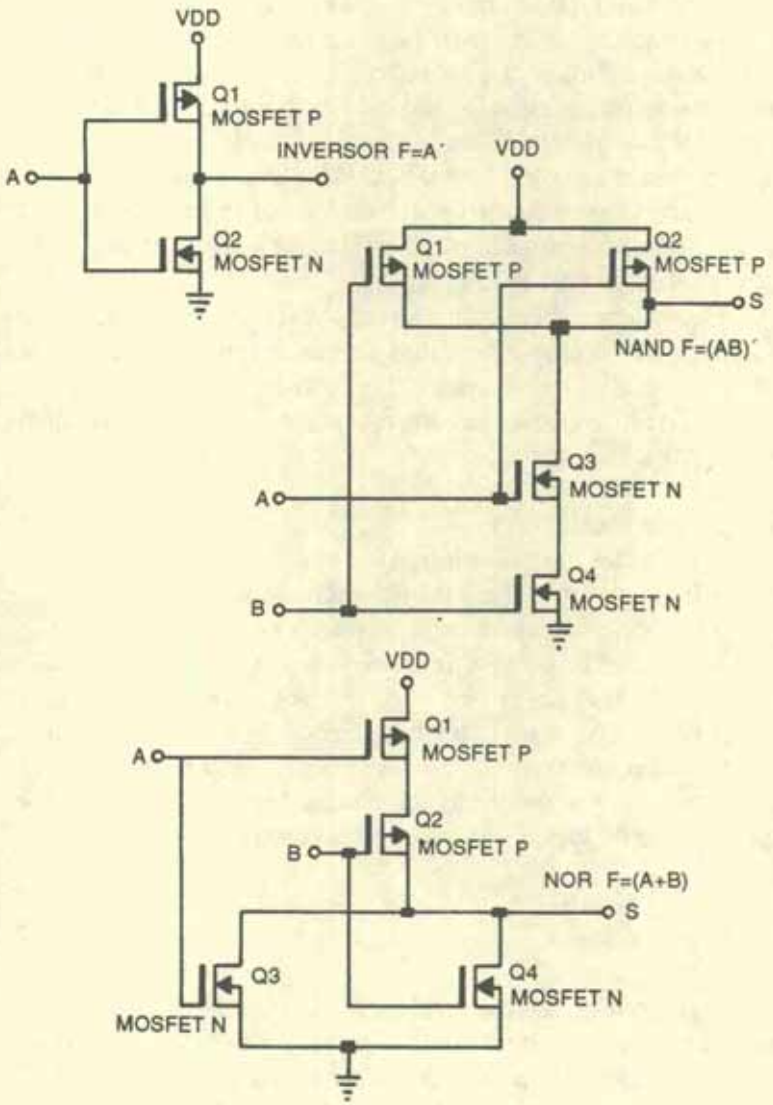


Figura 3.9.1. Circuitos lógicos con transistores CMOS

MOS con canal n en paralelo, de manera que cuando haya un 1 en la entrada la salida sea 0. También hay dos transistores canal p en serie, de manera que ambos deben estar conduciendo al mismo tiempo para que la salida sea igual a 1 sólo cuando ambas entradas estén aterrizadas.

Para la compuerta NAND hay dos transistores de canal n en serie hacia tierra y dos de canal p en paralelo hacia +V. Cuando alguna entrada está a tierra la salida es 1 y sólo cuando ambas entradas están en 1 la salida será 0.

El análisis de las compuertas NAND y NOR en cualquier familia lógica es muy importante porque a partir de ellas se puede construir cualquier sistema lógico con la complejidad que se desee. Lo único que se necesita es conectar de manera adecuada los bloques básicos.

Reglas para usar CMOS:

- Todas las entradas deben ir a tierra o a +Vcc.
- Evitar electricidad estática mientras se maneja.
- Es necesario almacenar los ICs en papel metálico.
- No debe usarse cautín para realizar uniones.
- Conectar primero los pins de Vcc y tierra.
- Las salidas de los CMOS no se deben poner a una sola salida. Dicho de otra manera, no se deben usar en lógica alambrada. Si se quiere hacer esto se deben usar CMOS especiales de lógica de tres estados como en los TTL.

3.10 Conclusiones

Saber cómo funcionan internamente los circuitos integrados es muy útil porque hace más fácil su manejo. Permite que se tomen en cuenta sus ventajas y desventajas, así como también saber la razón por la que no funciona algún diseño digital.

Al finalizar este tema se recomienda llevar a cabo las prácticas tres y cuatro, donde se experimenta con el uso de compuertas de colector abierto y la lógica de tres estados.

3.11 Ejercicios

Implementar las siguientes funciones con lógica DTL, DCTL, RTL, CMOS, NMOS y TTL.

1. f = a b'
2. f = a c d + b c d
3. f = a + b' c
4. f = a b + a' b' c
5. f = a b c' + d
6. f = a + b
7. f = c a' + c' b'

CAPÍTULO IV
Sistemas numéricos

Cuando se estudian circuitos digitales, un tema que no puede faltar es el de los sistemas numéricos, por su relación con aquéllos. Antes de aparecer las computadoras, el sistema numérico más importante era el decimal. En nuestros días, el sistema binario también es importante, lo mismo que otros que se relacionan mucho con éste.

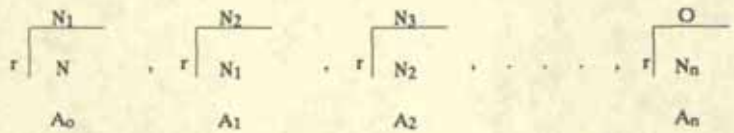
Si el lector ya está familiarizado con este tema, no hará falta que lo lea, o bien, bastará con un breve repaso.

4.1 Conversión entre bases

A menos que se especifique otra cosa, los números que se utilizan comúnmente están con base 10. Sin embargo, un número cualquiera N tiene una base s, y este número será igual a otro número M con una base r diferente a la del número N. Podemos decir entonces que:

$$N_s = M_r$$

Cualquier número N con base s se puede convertir a otro número M con base r mediante una secuencia de divisiones efectuadas en la base s. Los residuos de la división van a formar el nuevo número. La división será entre r, y los residuos serán menores que r (Ai < r):



Esto se puede escribir de la siguiente manera:

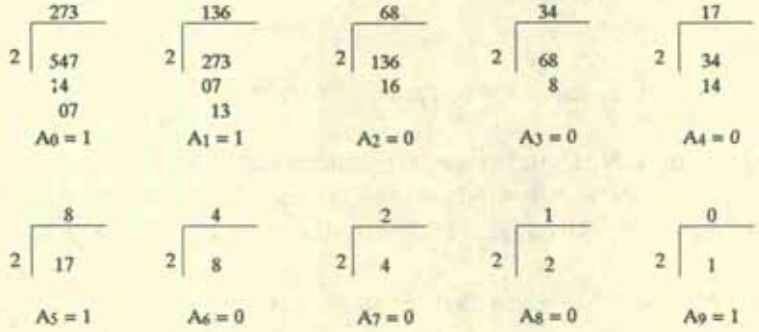
$$N = rN_1 + A_0$$
$$N_1 = rN_2 + A_1$$
$$N_2 = rN_3 + A_2$$
$$\vdots$$
$$N_n = 0\,r + A_n{}^3$$

También se puede expresar como sigue:

$$\begin{aligned} N &= r\,(rN_2 + A_1) + A_0 \\ &= r^2N_2 + rA_1 + A_0 \\ &= r^2\,(rN_3 + A_2) + rA_1 + A_0 \\ &= r^3N_3 + r^2A_2 + rA_1 + A_0 \\ &\vdots \\ &= r^n\,N_n + r^{n-1}A_{n-1} + \ldots + rA_1 + A_0 \end{aligned}$$
$$N = r^n\,A_n + r^{n-1}A_{n-1} + \ldots + rA_1 + A_0$$

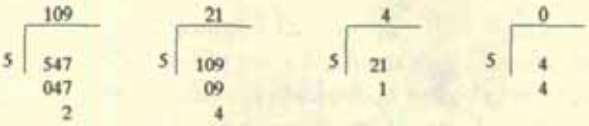
Ejemplo: Convertir 547_{10} a base 2 y a base 5:

A base 2:



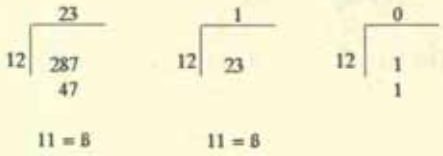
Por lo tanto, el número es 1000100011, empezando con A_9 y terminando con A_0 como el dígito menos significativo.

A base 5:



El número en base 5 es 4142. Nótese que en base dos el dígito 1 es el de más valor. En base cinco será el 4.

Ahora conviértase el número 376 de base 10 a base 12.



El número es 1ßß. En este caso, como el sistema de base 12 tiene más de 9 dígitos, se utiliza la letra alfa para el número 10 y beta para el número 11.

Conversión de números con partes fraccionarias

Un número N es igual a una parte entera y una parte fraccionaria, o sea que: $N = N_E + N_F$, donde N_E es igual a la parte entera del número y N_F es igual a la parte fraccionaria. Por lo tanto:

$$N = A_n R^n + \ldots + A_1 r + A_0 + A_{-1} r^{-1} + A_{-2} r^{-2} + \ldots$$

La parte entera del número se convierte a otra base como ya se explicó con anterioridad. La parte entera se va a convertir a otra base como sigue:

$$\text{Si } N_F = A_{-1} r^{-1} + A_{-2} r^{-2} + A_{-3} r^{-3} \ldots \text{ ecuación 1}$$

Donde A_{-n} es la parte entera del número $A_{-n} r^{-n}$.

Si se necesita determinar los coeficientes A_{-1}, A_{-2}, A_{-3}, . . ., etcétera, para la base r, como cada coeficiente es un entero, para encontrar A_{-1} es suficiente con que se multiplique la ecuación 1 por r.

$$rN_F = A_{-1} r^{-1} r + A_{-2} r^{-2} r + A_{-3} r^{-3} r + \ldots,$$

por lo tanto:

$$rN_F = A_{-1} + A_{-2} r^{-1} + A_{-3} r^{-2} + \ldots$$

Así se encuentra cuánto vale A_{-1}. Para encontrar A_{-2} se despeja A_{-2}:

$$r(rN_F - A_{-1}) = A_{-2} + A_{-3}r + \ldots$$

Este procedimiento se sigue hasta que se obtengan tantos coeficientes como se deseen. En ciertos números puede ser que el proceso nunca termine.

Ejemplo. Convertir el número 0.81 a base 2.

2 (0.81) = 1.62	A - 1 = 1
2 (0.62) = 1.24	A - 2 = 1
2 (0.24) = 0.48	A - 3 = 0
2 (0.48) = 0.96	A - 4 = 0
2 (0.96) = 1.92	A - 5 = 1
2 (0.92) = 1.84	A - 6 = 1
2 (0.84) = 1.68	A - 7 = 1

El número es aproximadamente igual a 0.1100111. Si se requiere de más exactitud se deberán sacar más coeficientes.

Ya se vio cómo convertir de una base a otra. Esto implica que puede haber un sistema numérico en cualquier base. A continuación se verán los sistemas más importantes, ya sea porque están relacionados con la base 2, o porque son muy comunes.

4.2 Sistema decimal

Durante muchos años se ha aprendido a contar, sumar, restar, multiplicar y dividir en un sistema numérico cuya base es 10. Este sistema tiene diez símbolos diferentes que pueden representar diferentes cantidades. Por ejemplo:

$$4532 = 4 \times 10^3 + 5 \times 10^2 + 3 \times 10^1 + 2 \times 10^0$$

Comúnmente se piensa que este sistema fue desarrollado porque el hombre tiene diez dedos. Como se está muy familiarizado con este sistema, es el sistema que se usa con mayor facilidad.

En materia de circuitos electrónicos hay dos condiciones o estados estables convenientes: encendido y apagado, ON y OFF. Un interruptor puede estar abierto o cerrado. Un transistor puede estar en corte o saturación. Por lo tanto, si para el ser humano el sistema en base 10 es el sistema numérico natural, para los circuitos electrónicos el sistema numérico natural es el sistema binario o sistema de base 2.

4.3 Sistema binario

El sistema binario tiene sólo dos números, el 1 y el 0. El sistema octal y el sistema hexadecimal están muy relacionados con la base 2 porque son potencias de dos. La reglas del sistema con base 10 se aplican a cualquier otro sistema. Primero, se deberán usar todos los dígitos de la base, luego se iniciará una nueva columna para representar cantidades más altas. Si el sistema numérico tiene más de los diez símbolos o guarismos que utilizamos en el sistema decimal (0, 1, 2, 3, 4, 5, 6, 7, 8 y 9) se pueden utilizar letras, como en la base hexadecimal utiliza las letras A,B,C,D, E y F para representar los números del diez al quince.

4.4 Sistemas con bases diferentes de diez y de dos

El sistema numérico con base cinco tiene cinco dígitos:

0, 1, 2, 3 y 4

La base cuatro tiene cuatro dígitos:

0, 1, 2 y 3

La base octal tiene ocho dígitos:

0, 1, 2, 3, 4, 5, 6 y 7

La base hexadecimal tiene diez y seis dígitos:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F

Como ya se dijo antes, la base 4, la base 8 y la base 16 están relacionadas con la base 2 pues $8 = 2^3$ y $16 = 2^4$. Por eso es muy fácil convertir números en base 2 a base 4, 8 y 16 directamente.

Conversión de base dos a base ocho

Si $8 = 2^3$, entonces cada dígito octal corresponde a tres dígitos binarios. Por lo tanto, el número binario:

10 011 100 101 010 111 001 100 110 100 001

se convierte directamente a la base 8 separando el número binario de tres en tres cifras de derecha a izquierda como sigue y encontrando el equivalente de cada número binario en la base octal:

10 011 100 101 010 111 001 100 110 100 001

2 3 4 5 2 7 1 4 6 4 1

Por lo tanto, el número en base 8 es 23452714641.

Conversión de base 2 a base 16

Si se quiere convertir este mismo número a la base hexadecimal bastaría con separar el número de cuatro en cuatro cifras. Puesto que $16 = 2^4$, cada dígito hexadecimal corresponde a cuatro dígitos binarios.

1001	1100	1010	1011	1001	1001	1010	0001
9	C	A	B	9	9	A	1

4.5 Aritmética con diferentes bases

Así como hay tablas de sumar con base 10, también se pueden tener tablas de sumar para cualquier otra base. A continuación se muestran las tablas de sumar y multiplicar en base dos, cuatro y ocho.

+	0	1
0	0	1
1	1	10

TABLA DE SUMAR BASE 2

x	0	1
0	0	0
1	0	1

TABLA DE MULTIPLICAR BASE 2

+	0	1	2	3
0	0	1	2	3
1	1	2	3	10
2	2	3	10	11
3	3	10	11	12

x	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	10	12
3	0	3	12	21

TABLAS PARA LA BASE 4

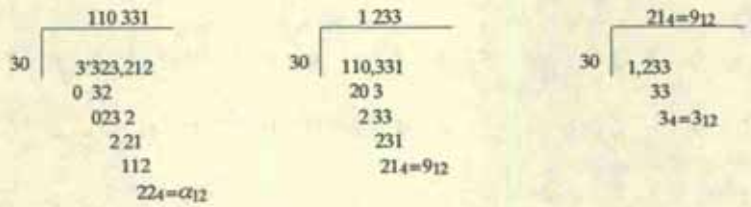
+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

x	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

TABLAS PARA LA BASE 8

Todas las reglas aritméticas se aplican igual que en la base diez. Ejemplo. Convertir el número 3'323,212 que está en base 4 a la base 12 directamente teniendo en cuenta que: cualquier número N con base s se puede convertir a otro número M con base r mediante una secuencia de divisiones efectuadas en la base s. Los residuos de la división van a formar el nuevo número. La división será entre r y los residuos serán menores que r (Ai < r).

En este caso se dividirá el número entre 12. Como las divisiones serán en la base 4 se deberá convertir el número a la base 4 y los residuos a la base 12, puesto que al efectuar las divisiones en base cuatro los residuos estarán en base cuatro. Por lo tanto se soluciona así:



El número en base 12 es 9,39α. Para demostrarlo se puede convertir 9,39α (base 12) a la base 10 y 3'323,212 (base 4) a la base 10 y verificar que ambos dan el mismo resultado.

939αen base 12 = 16 102 en base 10 = 3'323,212 en base 4

En el sistema de números decimales la multiplicación de un número por una potencia de diez sólo requiere una operación de corrimiento. Por ejemplo $4 \times 10^3 = 4{,}000$. Esto se aplica a cualquier otra base, por ejemplo:

1) $25_{10} \times 2^4 = (11001 \times 10000) = 110010000$
2) $31_5 \times 5^3 = 31000$

Ejercicios

Convertir los siguientes números de base 6 a la base 2 directamente:

a) 4532
b) 3421
c) 2643

Convertir los siguientes números de base 8 a base 2, base 4, y base 16:

a) 6572
b) 4571
c) 4721

Efectuar las siguientes operaciones en base 2, 4, y 12. Los números están en base 10.

a) 34 + 16
b) 14.85 - 11.32
c) 31/12
d) 23 x 7

4.6 Números complementarios y su aplicación

Cuando nosotros tenemos un número negativo en cualquier base, el número se resta igual que los números de base 10. Por ejemplo:

11001 - 10011 = 00110, o sea que 25 - 19 = 6

Estas operaciones se pueden implementar en los circuitos integrados de una computadora.

Complemento a 2. El complemento a 2 sirve para que en una computadora los circuitos empleados en la operación de suma se puedan usar también para restar.

Si A es un número negativo en base 2, entonces $A < 1$. Por definición el complemento a dos de A es:

$$A_C = 2 - A$$

De acuerdo con esta fórmula, el complemento a dos del número 0.11011000111 será

10.00000000000 -	2 -
0.11011000111 =	A =
1.00100111001	A_C

Regla para obtener el complemento a dos de un número menor que uno:

1) Escribir un 1 en el lugar del dígito correspondiente al 1 menos significativo.
2) Complementar los dígitos restantes (cambiando los ceros a unos y los unos a ceros).

Los números negativos en la forma de complemento a 2 se distinguirán siempre por un 1 inmediatamente a la izquierda del punto binario. Para los números positivos este dígito será 0. Así:

$$B - A$$

se puede expresar como

$$B + A_C = B + (2 - A) = (B - A) + 2,$$

si A < B, entonces

$$(B + A_C) > 2.$$

Este exceso aparecerá como un segundo bit significativo a la izquierda del punto binario. Sin embargo, se hará caso omiso de este dígito y se obtendrán resultados consistentes para cualquier combinación de signos y magnitudes de A y B. Si A > B, el resultado será un número negativo, expresado en la notación de complemento a dos. Este caso se distinguirá en el 1 que aparece a la izquierda del punto binario. Este 1 no va a aparecer jamás si A < B, pues el resultado es un número positivo.

Como ejemplo, efectuar las siguientes operaciones en aritmética de complemento a dos:

SIN COMPLEMENTO A DOS	CON COMPLEMENTO A DOS
0.1101—	0.1101 +
0.1001 =	1.0111 =
0.0100	10.0100 ← El 0 indica que el número es positivo.

0.1001—	Complemento	0.1001 +
0.1101 =		1.0011 =
—0.0100	—0.0100	= 1.1100 ← El 1 indica que el número es negativo. Por lo tanto, se le saca complemento.

A cualquier número en cualquier otra base se le puede sacar su complemento. Por ejemplo, el complemento de 10 de 3423 será 6577. El complemento a diez se obtiene al restar

10000 - 3423

Como ejercicio efectuar las siguientes restas utilizando el complemento a 10:

a) 4563 - 2345
b) 675.543 - 453.432
c) 6543.3 - 4322.3

Ahora efectuar las siguientes restas utilizando el complemento a 2:

a) 1000.001 - 1011.110
b) 11011 - 10011
c) .11000 - .01110

Código BCD *(Binario Codificado en Decimal)*

Todos los sistemas digitales en una computadora son binarios. Sin embargo, algunos efectúan operaciones aritméticas en el sistema decimal. Esto se logra haciendo que cada dígito decimal se represente individualmente mediante un código binario. Cada dígito decimal requiere cuatro bits binarios para su representación y se denomina código BCD. Este código se muestra a continuación:

Dígito decimal	Representación binaria
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Los números entran en una computadora en forma decimal. Al entrar se convierten a la forma binaria para su procesamiento, y este proceso se invierte al salir. En una computadora binaria el número 75 entraría como 1001011. En una computadora que utiliza el sistema BCD el 75 se convertirá en 0111 0101. La suma en una máquina BCD se llevará a cabo de la siguiente manera:

Decimal	BCD
37 +	0011 0111 +
24 =	0010 0100 =
61	0110 0001

La principal ventaja del sistema BCD es la simplicidad de la conversión I/O. Su desventaja es la complejidad del procesamiento aritmético.

El código BCD no es la única codificación posible para los dígitos decimales. Más adelante se analizan otros códigos que se emplean comúnmente.

4.7 Conclusiones

Al terminar este capítulo se estará en condiciones de manejar los sistemas numéricos con mucha habilidad, en especial con bases dos, ocho y dieciséis. Se recomienda finalizar con el último de los cinco ejercicios al final del presente capítulo, con el propósito de aplicar los conocimientos adquiridos en un ALU, puesto que éste es uno de los circuitos integrados que manejan directamente el sistema binario. A la salida del circuito se puede agregar un decodificador y un display para tener una lectura en números decimales. Esto implicará realizar las prácticas seis y siete.

4.8 Ejercicios

1. Convertir el número hexadecimal A2C5D3 a su equivalente octal y a su equivalente binario. Indicar cuántos bits componen este número.

2. Convertir a hexadecimal y octal los siguientes números binarios:

   a) 101110001.101110001
   b) 111011101110.001100110011

3. Convertir los siguientes números decimales a binario:

   a) 231.25
   b) 37.625
   c) 46.724

4. Efectuar las siguientes restas utilizando el complemento del sustraendo:

a) 1100110 - 1001010
b) 8756 - 3452

5. Representar el número decimal 3456 en BCD, código reflejado y en ASCII.

CAPÍTULO V
Método gráfico para la minimización de las funciones de Boole

Cuando se diseña un sistema digital ya no es preciso preocuparse de su construcción interna, sino sólo de sus propiedades lógicas externas. Por otro lado, ya se vio cómo diseñar un circuito utilizando las leyes del álgebra de Boole. Ahora se verá otro método de diseño, que es el método gráfico. Se trata de un método manual sencillo y útil.

5.1 Forma estándar de las funciones

Para poder minimizar funciones con el método gráfico, se tienen que estandarizar las funciones, es decir, expresarlas de una forma común. La estandarización permite facilitar el procedimiento de simplificación.

La estandarización significa la representación de las funciones como circuitos de dos niveles. Ver figura 2.6.1 (a). El primero está formado por las compuertas AND y el segundo está formado por una compuerta OR de cuatro entradas.

En la minimización por el método gráfico la función puede representarse como una suma de productos:

$$a\,b\,c' + a'\,b\,c + a'\,b'\,c' + a\,b\,c,$$

o como un producto de sumas:

$$(a + b + c)\,(a' + b' + c')\,(a + b + c')\,(a' + b + c)$$

Se habrá notado que en la suma de productos, cada uno de ellos contiene todas las variables; en el producto de sumas, cada suma contiene todas las variables. Todas las funciones se pueden expresar como una suma de productos o como un producto de sumas. Ejemplo. exprese la siguiente función como una suma de productos:

$$\begin{aligned}
F(A, B, C, D) &= D\,(A' + B) + B'\,D \\
&= A'\,D + B\,D + B'\,D \\
&= A'\,D\,(B + B') + B\,D\,(A + A') \\
&\quad + B'\,D\,(A + A') \\
&= A'\,B\,D\,(C + C') + A'\,B'\,D\,(C + C') \\
&\quad + A\,B\,D\,(C + C') + A'\,B\,D\,(C + C') \\
&\quad + A\,B'\,D\,(C + C') + A'\,B'\,D\,(C + C') \\
&= A'\,B\,C\,D + A'\,B\,C'\,D + A'\,B'\,C\,D \\
&\quad + A'\,B'\,C'\,D + A\,B\,C\,D + A\,B\,C'\,D \\
&\quad + A'\,B\,C\,D + A'\,B\,C'\,D + A\,B'\,C\,D \\
&\quad + A\,B'\,C'\,D + A'\,B'\,C\,D + A'\,B'\,C'\,D
\end{aligned}$$

A continuación se eliminan los términos repetidos, teniendo en cuenta que cada uno de los términos representa un número binario donde las letras sin negar representan un 1 y las negadas un 0. Las letras se ponen en orden, de menor a mayor. Por lo tanto:

$$\begin{aligned}
&= A'\,B'\,C'\,D + A'\,B'\,C\,D + A'\,B\,C'\,D \\
&\quad + A'\,B\,C\,D + A\,B'\,C'\,D + A\,B'\,C\,D \\
&\quad + A\,B\,C\,D + A\,B\,C'\,D
\end{aligned}$$

Estos números representan el 1, el 3, el 5, el 7, el 9, el 11, el 13 y el 15.

Ahora representar el mismo ejemplo como un producto de sumas:

$$
\begin{aligned}
F(A,B,C,D) &= D\,(A' + B) + B'\,D \\
&= (D\,(A' + B) + B')\,(D\,(A' + B) + D) \\
&= (B' + D)\,(B' + B + A')\,(D + D) \\
&\quad (A' + B + D) \\
&= (B' + D)\,(D)\,(A' + B + D)
\end{aligned}
$$

Ya que tenemos sólo productos, con la ayuda del álgebra de Boole se obtendrá el producto de sumas, donde cada suma contenga todas las variables.

$$
\begin{aligned}
&= (B' + D + A\,A')\,(D + A\,A') \\
&\quad (A' + B + D + C\,C') \\
&= (B' + D + A)\,(B' + D + A')\,(D + A) \\
&\quad (D + A')\,(A' + B + C + D) \\
&\quad (A' + B + C' + D) \\
&= (B' + D + A + C\,C')\,(B' + D + A' + C\,C') \\
&\quad (D + A + C\,C')\,(D + A' + C\,C') \\
&\quad (A' + B + C + D)\,(A' + B + C' + D) \\
&= (B' + D + A + C)\,(B' + D + A + C') \\
&\quad (B' + D + A' + C)\,(B' + D + A' + C') \\
&\quad (D + A + C)\,(D + A + C')\,(D + A' + C) \\
&\quad (D + A' + C')\,(A' + B + C + D) \\
&\quad (A' + B + C' + D) \\
&= (A + B' + C + D)\,(A + B' + C' + D) \\
&\quad (A' + B' + C + D)\,(A' + B' + C' + D) \\
&\quad (D + A + C + B\,B')\,(D + A + C' + B\,B') \\
&\quad (D + A' + C + B\,B')\,(D + A' + C' + B\,B') \\
&\quad (A' + B + C + D)\,(A' + B + C' + D)
\end{aligned}
$$

$$= (A + B' + C + D)(A + B' + C' + D)$$
$$(A' + B' + C + D)(A' + B' + C' + D)$$
$$(A + B + C + D)(A + B' + C + D)$$
$$(A + B + C' + D)(A + B' + C' + D)$$
$$(A' + B + C + D)(A' + B' + C + D)$$
$$(A' + B + C' + D)(A' + B' + C' + D)$$
$$(A' + B + C + D)(A' + B + C' + D)$$

En seguida se acomodan los términos, empezando por el más significativo.

$$= (A + B + C + D)(A + B + C' + D)$$
$$(A + B' + C + D)(A + B' + C' + D)$$
$$(A' + B + C + D)(A' + B + C' + D)$$
$$(A' + B' + C + D)(A' + B' + C' + D)$$

Estos números representan al 0, 2, 4, 6, 8, 10, 12 y 14.

En el producto de sumas, la suma de A + B + C + D representa el 0, porque cuando A = 0, B = 0, C = 0 y D = 0; entonces A + B + C + D = 0. Como la suma es parte de un producto punto (AND), toda la función será igual a 0. Será 1 sólo cuando se presenten los números 1, 3, 5, 7, 9, 11, 13 y 15. Este es el resultado esperado pues se pretende que la función inicial sea igual a 1 si y sólo si F = D (A' + B) + B' D.

En el ejemplo anterior se demostró que cualquier función se puede representar como una suma de productos o como un producto de sumas. Haremos otro ejemplo más sencillo para entender por qué en el producto de sumas se toman los términos con las variables complementadas. (A + B + C + D = 0).

Ejemplo: se desea obtener un circuito que detecte los números en binario que tengan dos unos del 0 al 8. Es decir, que el circuito deberá dar un 1 cuando a la entrada del circuito haya un 3, un 5 o un seis. La tabla de verdad para el circuito es la siguiente:

ABC	F
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	0

La función de salida F es igual a la siguiente suma de productos: F = A' B C + A B' C + A B C'; de modo que si a la entrada se tiene un 011, un 101 o un 110, entonces F será igual a 1. En cualquier otro caso F será igual a 0.

La función de salida F es igual al siguiente producto de sumas:

$$F = (A + B + C)(A + B + C')(A + B' + C)(A' + B + C)(A' + B' + C')$$

Si en la entrada se tiene un 011, un 101 o un 110, entonces F será igual a 1. En cualquier otro caso F será igual a 0.

5.2 Mintérminos y maxtérminos

En la suma de productos, a cada producto se le llama mintérmino, nombre aplicado a cada uno de los términos AND de la suma. Como se habrá notado, el circuito que representa una suma estándar de productos es una serie de compuertas AND cuyas salidas están conectadas a una compuerta OR.

Por otra parte, en el producto de sumas se le llama maxtérmino a cada suma, nombre aplicado a cada uno de los términos OR del producto. El circuito que representa el producto de maxtérminos es una serie de compuertas OR cuyas salidas están conectadas a una compuerta AND. La tabla 5.2.1 muestra los mintérminos y maxtérminos para las tres variables binarias.

	Mintérminos		Máxtérminos	
a b c	Término	Notación	Término	Notación
0 0 0	a' b' c'	m_0	a + b +c	M_0
0 0 1	a' b' c	m_1	a + b +c'	M_1
0 1 0	a' b c'	m_2	a + b'+c	M_2
0 1 1	a' b c	m_3	a + b'+c'	M_3
1 0 0	a b' c'	m_4	a'+ b +c	M_4
1 0 1	a b' c	m_5	a'+ b +c'	M_5
1 1 0	a b c'	m_6	a'+ b'+c	M_6
1 1 1	a b c	m_7	a'+ b'+c'	M_7

Véase un ejemplo con todas las formas para representar una suma de mintérminos y un producto de maxtérminos. Se tiene la siguiente función de tres variables con dos salidas. Exprésela como: a) suma de mintérminos, y b) producto de maxtérminos:

a b c	f_1	f_2
0 0 0	0	0
0 0 1	1	0
0 1 0	0	0
0 1 1	0	1
1 0 0	1	0
1 0 1	0	1
1 1 0	0	1
1 1 1	1	1

Suma de mintérminos

$f_1 = a'\,b'\,c + a\,b'\,c' + a\,b\,c$

$f_1 = m_1 + m_4 + m_7$

$f_1(a, b, c) = \sum m\,(1, 4, 7)$

$f_2 = a'\,b\,c + a\,b'\,c + a\,b\,c' + a\,b\,c$

$f_2 = m_3 + m_5 + m_6 + m_7$

$f_2(a, b, c) = \sum m\,(3, 5, 6, 7)$

Producto de maxtérminos

$f_1 = a\,b\,c + a\,b'\,c + a\,b'\,c' + a'\,b\,c' + a'\,b'\,c$

$f_1 = M_0 \bullet M_2 \bullet M_3 \bullet M_5 \bullet M_6$

$f_1(a, b, c) = \Pi M\,(0, 2, 3, 5, 6)$

$f_2 = a\,b\,c + a\,b\,c' + a\,b'\,c + a'\,b\,c$

$f_2 = M0 \bullet M1 \bullet M2 \bullet M4$

$f_2 = \Pi M\,(0, 1, 2, 4)$

5.3 El mapa de Karnaugh

Ya se sabe como representar una función en forma estándar. Por lo regular uno recibirá la función en forma estándar para minimizarla. El método de mapas es un procedimiento simple para minimizar las funciones de Boole. Este método se considera como el método gráfico de representación de tablas de verdad y es conocido como diagrama de Veitch o mapa de Karnaugh. El mapa es un diagrama compuesto por cuadros. Cada uno representa un mintérmino. De este se derivan expresiones algebraicas simples, que pueden ser una suma de productos o un producto de sumas. Se obtiene así una función de dos niveles con un número mínimo de literales. Se verán primero los mapas y luego se verá cómo utilizarlos.

Mapas de dos variables

La figura 5.3.1 muestra un mapa de dos variables. Hay cuatro mintérminos para dos variables. Por lo tanto, el mapa consta de cuatro cuadros, uno para cada mintérmino.

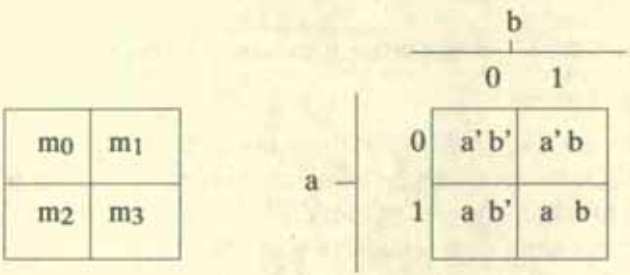


Figura 5.3.1. Mapa de Karnaugh para dos variables

Mapas de tres variables

La figura 5.3.2 muestra un mapa de tres variables. Hay ocho mintérminos para tres variables; por lo tanto el mapa consta de ocho cuadros:

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

		b c			
		0 0	0 1	1 1	1 0
a	0	a'b'c'	a'b'c	a'b c	a'b c'
	1	a b'c'	a b'c	a b c	a b c'

Figura 5.3.2. Mapa de Karnaugh para tres variables

Obsérvese que los mintérminos no están arreglados en una secuencia binaria, sino en una secuencia en la que la diferencia de un número a otro es el cambio de un solo bit. Hay un código que representa esta secuencia y es el código reflejado que se muestra a continuación para ocho bits. Se le llama código reflejado porque se forma a partir de "reflejar" los bits en un espejo imaginario.

1. Se inicia con los dígitos naturales del código binario: 0 y 1.
2. Se pone el espejo:

IMAGEN	ESPEJO
0	1
1	0

3. A los bits no reflejados se les antepone un 0 y a la imagen un 1.

0	0
0	1
1	1
1	0

4. Se cambia de lugar el espejo y se repite el paso tres.

0	0	0
0	0	1
0	1	1
0	1	0
1	1	0
1	1	1
1	0	1
1	0	0

Nótese que la característica de esta secuencia es que sólo un bit cambia de un número a otro. Cuando sólo un bit cambia de un número a otro se dice que hay adyacencia. En el mapa de Karnaugh, entre los cuadros existe adyacencia. Esta propiedad básica es muy importante pues cualquier cuadro adyacente a otro difiere sólo en una variable. Por ejemplo, el cuadro donde se encuentra el mintérmino 0 es adyacente al cuadro donde se encuentra el mintérmino 4. El 0 es igual **a: a' b'c'** y el 4 es igual **a: a b'c'**; por tanto, ambos mintérminos difieren sólo en la variable a. Si una función f es igual a : **f = a' b'c' + a b'c'**, si se minimiza esa función dará por resultado **f = b'c'**. Se

puede definir entonces que cualquier combinación de dos mintérminos adyacentes da por resultado una expresión de dos literales.

Si una función f es: **f=a'b'c' + a'b c' + a b'c' + a b c'**; o, escrito de otra manera: **f = m_0 + m_2 + m_4 + m_6**; entre m_0 y m_2 existe adyacencia; m_2 a su vez tiene adyacencia con el mintérmino m_6. Este tiene también con m_4, y m_4 con m_0. Entre los cuatro hay relación. Si se minimiza esta función con álgebra de Boole se obtendrá que f = c'. Por lo tanto, es posible afirmar que cualquier combinación de cuatro mintérminos adyacentes resulta en una expresión de una sola literal.

Mapa de cuatro variables

La figura 5.3.3 muestra un mapa de cuatro variables.

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

a b \ c d	0 0	0 1	1 1	1 0
0 0	A'B'C'D'	A'B'C'D	A'B'C D	A'B'C D'
0 1	A'B C'D'	A'B C'D	A'B C D	A'B C D'
1 1	A B C'D'	A B C'D	A B C D	A B C D'
1 0	A B'C'D'	A B'C'D	A B'C D	A B'C D'

Figura 5.3.3. Mapa de cuatro variables

Como se notará, se mantiene la adyacencia entre los cuadros de cuatro variables. En los mapas de Karnaugh, entre los cuadros siempre deberá existir adyacencia.

Para el mapa de cuatro variables se puede establecer que:

1. Un cuadro representa un mintérmino. Se obtiene un término de cuatro literales.
2. Dos cuadros adyacentes representan un término de tres literales.
3. Cuatro cuadros adyacentes representan un término de dos literales.
4. Ocho cuadros adyacentes representan un término de una literal.
5. 16 cuadros adyacentes representan la función igual a 1.

Mapa de cinco variables

Véase la figura 5.3.4 que esquematiza un mapa de cinco variables:

Figura 5.3.4. Mapa de cinco variables. La variable A representa el bit más significativo

Mapa de seis variables

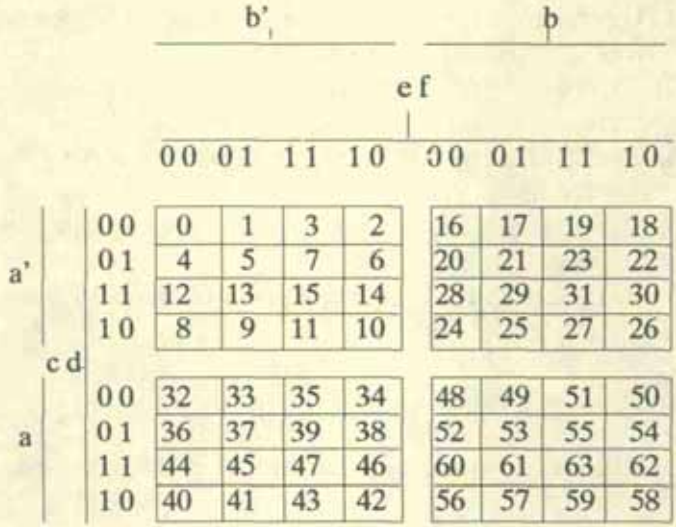


Figura 5.3.5. Mapa de seis variables

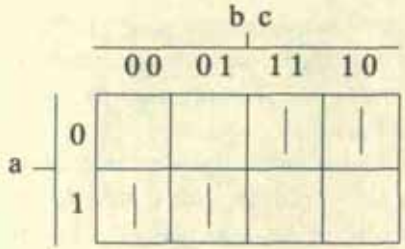
5.4 Minimización de sumas de productos

En seguida se minimizarán las funciones para obtener sumas de productos. Esto se verá con ejemplos para dos, tres, cuatro y cinco variables.

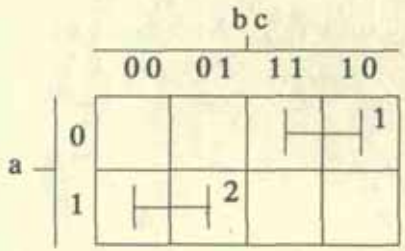
Ejemplo. Simplifique la función:

$$F(a, b, c) = a' b c + a' b c' + a b' c' + a b' c$$

Primero se localizan los mintérminos en el mapa de Karnaugh. Los mintérminos se representan con un 1 en el mapa. Los maxtérminos con un 0. Para la minimización de sumas de productos sólo se trabaja con mintérminos.



A continuación se forman grupos con los cuadros adyacentes. En este caso se formarán dos grupos.



El grupo uno está formado por los mintérminos 2 y 3; el grupo dos por los mintérminos 4 y 5. El mintérmino 2 es igual a a' b c' y el mintérmino 3 es igual a a' b c. Ambos son iguales, excepto por c, que en 2 es c' y en 3 es c. Esta es la variable que se elimina,

quedando únicamente a' b en la minimización. El mintérmino 4 y el mintérmino 5 son iguales excepto por la variable c; en el mintérmino 4 es c' y en el 5 es c. Esta es la variable que se elimina, quedando a b' en la minimización. El resultado es:

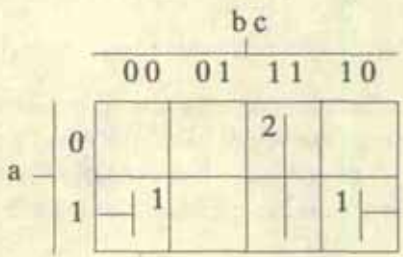
$$F(a, b, c) = a'b + ab'$$

Pasos para minimizar por el método gráfico, obteniendo una suma de productos:

1. Se localizan los mintérminos en el mapa de Karnaugh.
2. Se forman grupos con los mintérminos adyacentes, (se etiqueta cada grupo con un número). Se forma un número mínimo de grupos.
3. Se analizan los mintérminos de cada grupo, donde se eliminan las variables que cambian y sólo quedan las variables que permanecen igual. Hay que recordar que cualquier combinación de dos mintérminos adyacentes da por resultado una expresión de dos literales, y cualquier combinación de cuatro mintérminos adyacentes resulta en una expresión de una sola literal.

Ejemplo. Simplificar $F(a, b, c) = a'bc + ab'c' + abc + abc'$

Pasos uno y dos:

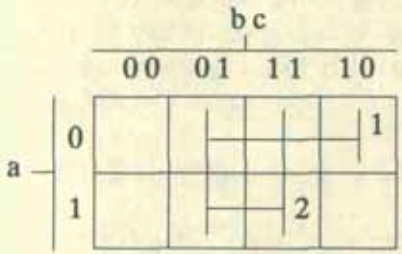


Paso tres: en el grupo uno se elimina b, y en el grupo dos se elimina a, quedando:

$$F(a, b, c) = a\,c' + b\,c$$

Ejemplo. Simplificar la función:

$$F = a'\,c + a'\,b + a\,b'\,c + b\,c$$

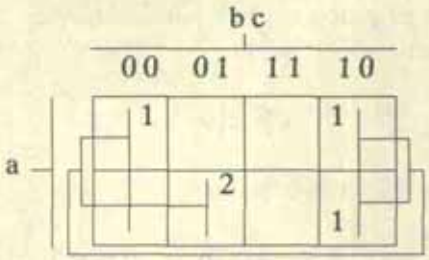


Puede haber intersección entre dos grupos, como se ve en este ejemplo donde el mintérmino 3 pertenece al grupo uno y al grupo dos. El grupo uno está formado por los mintérminos 2 y 3, y el grupo dos por los mintérminos 1, 3, 5 y 7. Para el grupo uno la variable que cambia es c que, por lo tanto, se elimina. Es preciso recordar que cualquier combinación de cuatro mintérminos adyacentes resulta en una expresión de una sola literal. Por ello, en el grupo dos las variables que cambian son a y b, que se eliminan, quedando sólo c después de la minimización. Finalmente:

$$F = a'\,b + c$$

Ejemplo . Simplificar la función:

$$F = \Sigma m\,(0, 2, 4, 5, 6)$$

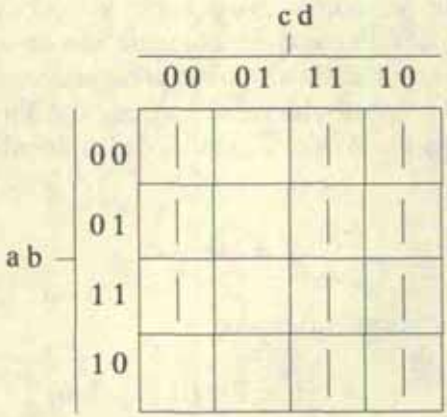


Al grupo uno pertenecen los mintérminos 0, 2, 4 y 6. Al grupo dos pertenecen los mintérminos 4 y 5. Esta vez, el 4 pertenece a ambos grupos. En el grupo uno las variables que cambian son a y b, quedando sólo c'. En el grupo dos la variable que cambia es c, quedando a b'. Finalmente: F = a b' + c'

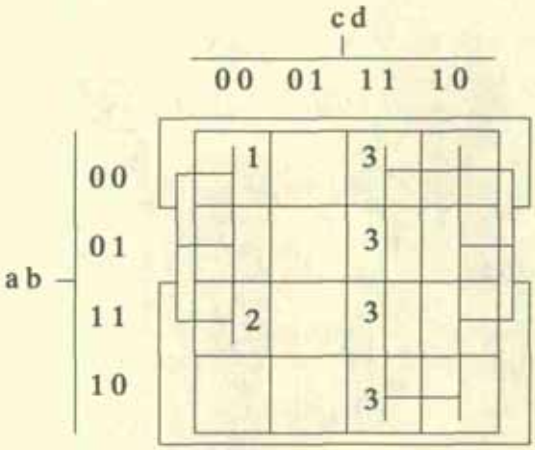
Ejemplo. Simplificar la siguiente función:

$$F = m_0 + m_2 + m_3 + m_4 + m_6 + m_7 + m_{10} + m_{11} + m_{12} + m_{14} + m_{15}$$

Paso 1. Se localizan los mintérminos:



Paso 2. Se hace el agrupamiento:



Paso 3. Se minimiza:

Al grupo uno pertenecen los mintérminos 0, 2, 4 y 6.
Al grupo dos pertenecen los mintérminos 4, 6, 12 y 14.
Al grupo tres pertenecen los mintérminos 2, 3, 6, 7, 10, 11, 14 y 15.
El grupo uno queda como a' d'.
El grupo dos queda como b d'.
El grupo tres queda como c.

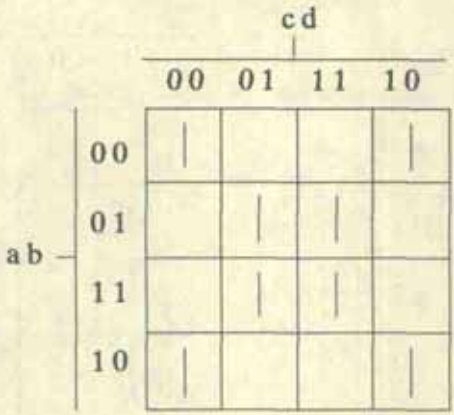
Por lo tanto, la función minimizada es:

$$F = a' d' + b d' + c$$

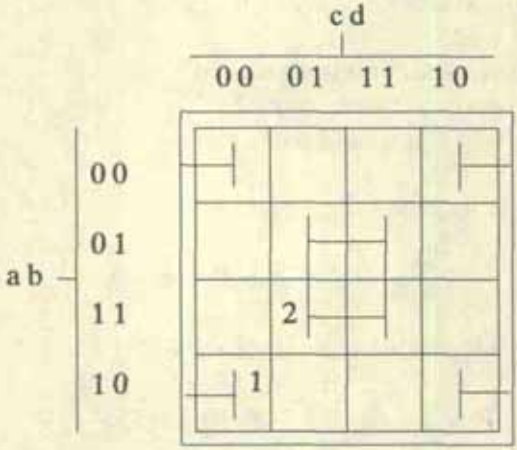
Ejemplo . Simplificar la siguiente función:

$$F = \sum m\ (0, 2, 5, 7, 8, 10, 13, 15)$$

Paso 1. Se localizan los mintérminos:



Paso 2. Se hace el agrupamiento:



Paso 3. Se minimiza:

Al grupo uno pertenecen los mintérminos 0, 2, 8 y 10.
Al grupo dos pertenecen los mintérminos 5, 7, 13 y 15.
El grupo uno queda como b' d'.
El grupo dos queda como b d.

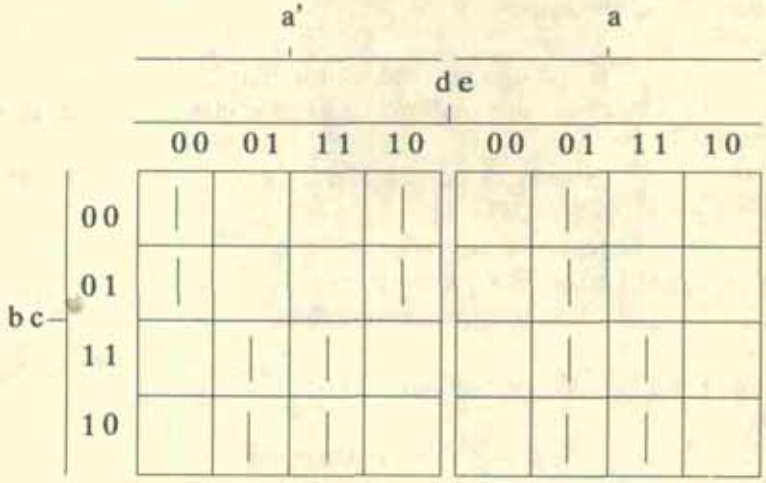
Por lo tanto, la función minimizada es:

$$F = b' d' + b d$$

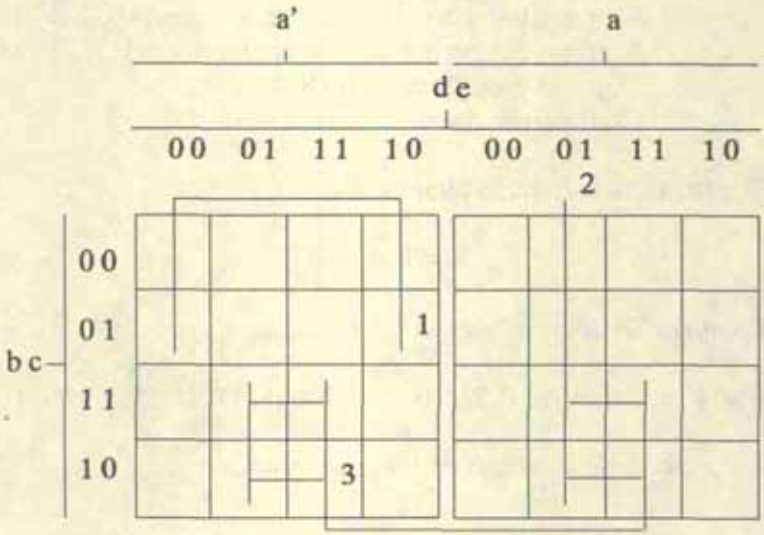
Ejemplo. Simplificar la siguiente función:

$F(a, b, c, d, e) = \sum m(0, 2, 4, 6, 9, 11, 13, 15, 17, 21, 25, 27, 29, 31)$

Paso 1. Se localizan los mintérminos.



Paso 2. Se hace el agrupamiento:



Paso 3. Se minimiza:

Al grupo uno pertenecen los mintérminos 0, 2, 4 y 6.
Al grupo dos pertenecen los mintérminos 17, 21, 25 y 29.
Al grupo tres pertenecen los mintérminos 9, 11, 13, 15, 25, 27, 29 y 31.
El grupo uno queda como a' b' e'.
El grupo dos queda como a d' e.
El grupo tres queda como b e.

Por lo tanto, la función minimizada es:

$$F = a' b' e' + a d' e + b e$$

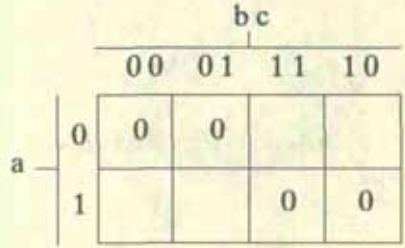
5.5 Minimización de productos de sumas

Todos los ejemplos anteriores se repetirán, pero ahora se resolverán por el método de producto de sumas. Algunas veces será más fácil minimizar utilizando la suma de productos; otras veces será más fácil utilizar el método de producto de sumas. En la práctica, si no se especifica otra cosa se podrá utilizar el método que se desee. Ahora se localizarán los maxtérminos en lugar de los mintérminos.

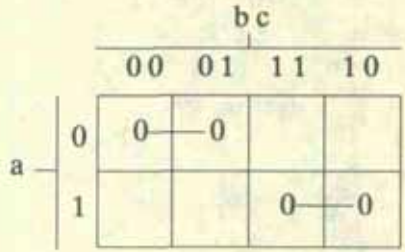
Ejemplo. Simplifique la función:

$$F(a, b, c) = a'\,b\,c + a'\,b\,c' + a\,b'\,c' + a\,b'\,c$$

Primero se localizan los maxtérminos en el mapa de Karnaugh y se representan con un 0 en el mapa. Los maxtérminos son todos los términos que no aparecen en la función que se pide que se simplifique:



En seguida se forman grupos con los cuadros adyacentes. En este caso se formarán dos grupos.



El grupo uno está formado por los maxtérminos 0 y 1, y el grupo dos por los maxtérminos 6 y 7. El maxtérmino 0 es igual a a + b + c y el maxtérmino 1 es igual a a + b + c'. Ambos son iguales excepto por c, que en 0 es c y en 1 es c'. Esta es la variable que se elimina, quedando únicamente a+b en la minimización.

El maxtérmino 6 y el maxtérmino 7 son iguales excepto por la variable c, donde en el maxtérmino 6 es c' y en 7 es c. Esta es la variable que se elimina, quedando a' + b' en la minimización. Por lo tanto, el resultado es:

$$F(a, b, c) = (a + b)(a' + b')$$

Si se retoma el ejemplo que pedía minimizar la función, se tendrá:

$$F(a, b, c) = a'bc + a'bc' + ab'c' + ab'c, \text{ y}$$
$$F = \sum m(2, 3, 4, 5)$$

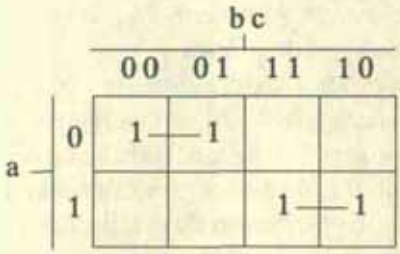
Por lo tanto,

$$F' = \sum m(0, 1, 6, 7)$$

Se puede obtener a partir de la función F' la función F utilizando el método de mapas.

a \ bc	00	01	11	10
0	1	1		
1			1	1

En seguida se forman grupos con los cuadros adyacentes. En este caso se formarán dos grupos.



La función minimizada sería:

$$F' = a'\,b' + a\,b$$

donde F = F' '. Por lo tanto:

$$F'' = ((a'\,b') + (a\,b))'$$

de manera que:

$$F = (a + b)\,(a' + b'),$$

que es exactamente el mismo resultado que se acaba de obtener al minimizar por el método de producto de sumas. Por tanto, se puede decir que:

$$F = (a + b)\,(a' + b') = a'\,b + a\,b'$$

La minimización prácticamente es igual, sólo que se obtienen sumas en lugar de productos.

Pasos para minimizar por el método gráfico obteniendo un producto de sumas:

1. Se localizan los maxtérminos en el mapa de Karnaugh.
2. Se forman grupos con los maxtérminos adyacentes, etiquetando a cada grupo con un número. Se formarán un número mínimo de grupos.
3. Se analizan los maxtérminos de cada grupo, donde se eliminan las variables que cambian y sólo quedan las variables que permanecen. Habrá que recordar que cualquier combinación de dos maxtérminos adyacentes da por resultado una expresión de dos literales, y que cualquier combinación de cuatro maxtérminos adyacentes resulta en una expresión de una sola literal.

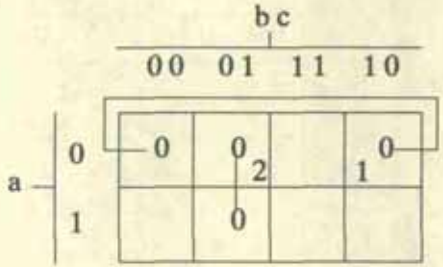
Ejemplo. Simplificar la función:

$$F(a, b, c) = \Pi M(0, 1, 2, 5)$$

Paso 1. Se localizan los maxtérminos:

a \ bc	00	01	11	10
0	0	0		0
1		0		

Paso 2. Se hace el agrupamiento:



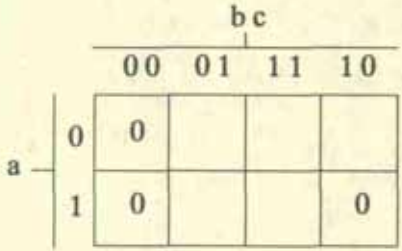
Paso 3. En el grupo uno se elimina b y en el grupo dos se elimina a, quedando:

$$F(a,b,c) = (a + c)(b + c')$$

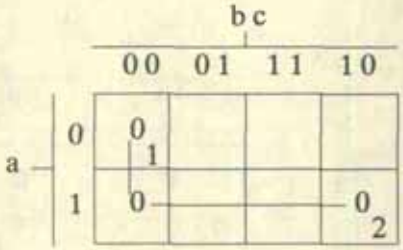
Ejemplo. Simplificar la función:

$$F = \Pi M\ (0, 4, 6)$$

Paso 1. Se localizan los maxtérminos:



Paso 2. Se hace el agrupamiento:



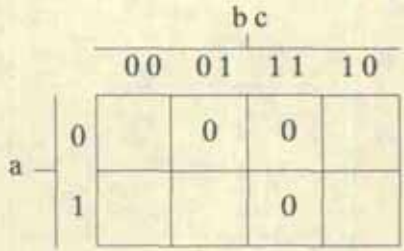
Puede haber intersección entre dos grupos, como se ve en este ejemplo donde el maxtérmino 4 pertenece a los grupos uno y dos. El grupo uno está formado por los maxtérminos 0 y 4, y el grupo dos lo está por los maxtérminos 4 y 6. Para el grupo uno la variable que cambia es a, que se elimina. En el grupo dos la variable que cambia es b, quedando con la minimización:

$$F = (a + b)(a' + c)$$

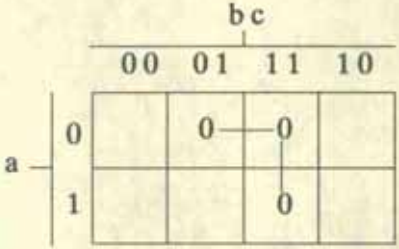
Ejemplo. Simplificar la función:

$$F = \Pi M\,(1, 3, 7)$$

Paso 1. Se localizan los maxtérminos.



Paso 2. Se hace el agrupamiento.



Al grupo uno pertenecen los maxtérminos 1 y 3. Al grupo dos pertenecen los maxtérminos 3 y 7. El maxtérmino 3 pertenece a ambos grupos. En el grupo uno la variable que cambia es b, quedando a + c'. En el grupo dos la variable que cambia es a, quedando b' + c'. Finalmente:

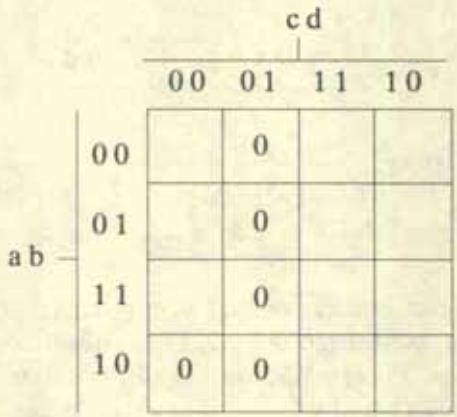
$$F = (a + c')(b' + c') = a\,b' + c'$$

Nótese que el resultado es el mismo que se obtuvo en el ejemplo anterior.

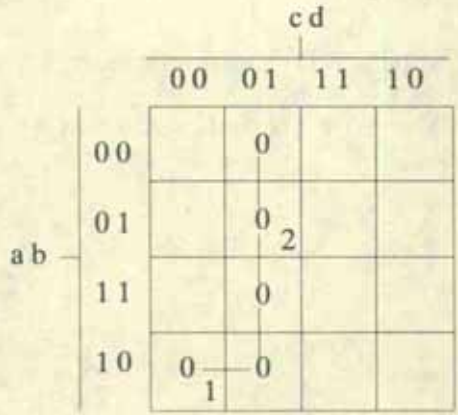
Ejemplo. Simplificar la siguiente función:

$$F = M_1 \bullet M_5 \bullet M_8 \bullet M_9 \bullet M_{13}$$

Paso 1. Se localizan los maxtérminos:



Paso 2. Se hace el agrupamiento:



Paso 3. Se minimiza:

Al grupo uno pertenecen los maxtérminos 8 y 9.
Al grupo dos pertenecen los maxtérminos 1, 5, 9 y 13.
El grupo uno queda como a' + b + c.
El grupo dos queda como c + d'.

La función minimizada es:

$$F = (a' + b + c)(c + d') = a' d' + b d' + c$$

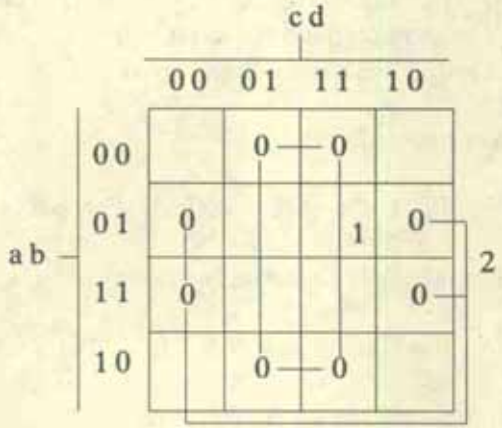
Ejemplo. Simplificar la siguiente función:

$$F = \Pi M\,(1, 3, 4, 6, 9, 11, 12, 14)$$

Paso 1. Se localizan los maxtérminos:

ab \ cd	00	01	11	10
00		0	0	
01	0			0
11	0			0
10		0	0	

Paso 2. Se hace el agrupamiento:



Paso 3. Se minimiza:

Al grupo uno pertenecen los maxtérminos 1, 3, 9 y 11.
Al grupo dos pertenecen los maxtérminos 4, 6, 12 y 14.
El grupo uno queda como b + d'.
El grupo dos queda como b'+ d.

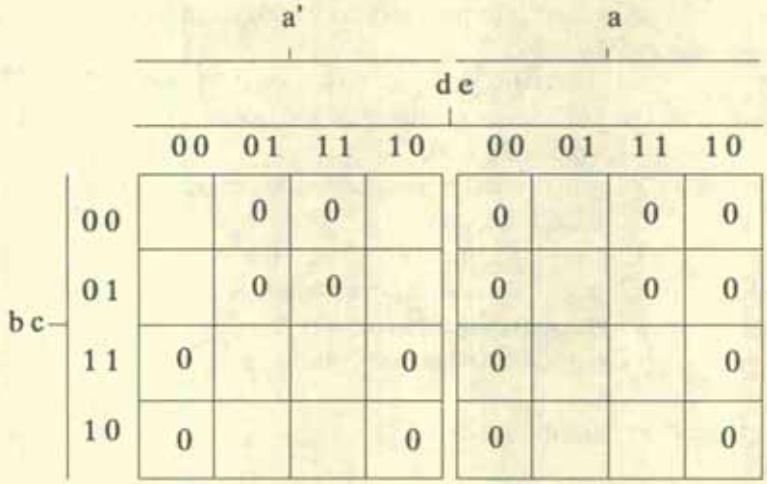
La función minimizada es:

$$F = (b + d')(b' + d) = b'd' + bd$$

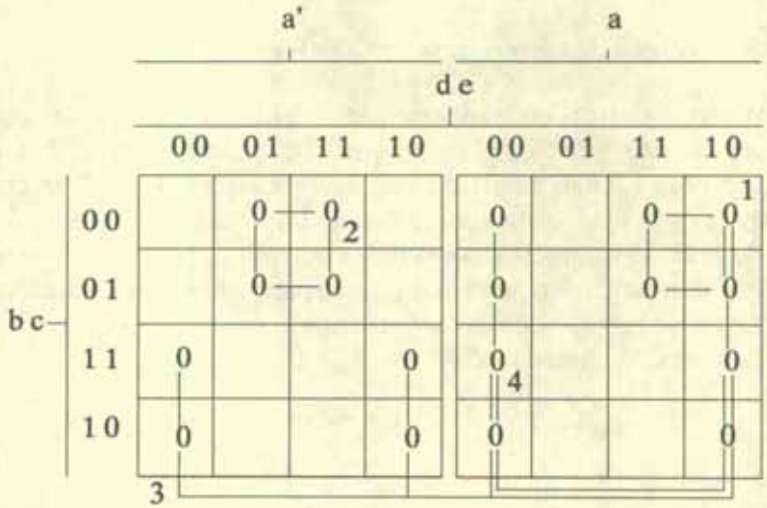
Ejemplo. Simplificar la siguiente función:

$$F(a, b, c, d, e) = \Pi M\ (1, 3, 5, 7, 8, 10, 12, 14, 16, 18, 19, 20, 22, 23, 24, 26, 28, 30)$$

Paso 1. Se localizan los maxtérminos:



Paso 2. Se hace el agrupamiento:



Paso 3. Se minimiza:

Al grupo uno pertenecen los maxtérminos 18, 19, 22 y 23.
Al grupo dos pertenecen los maxtérminos 1, 3, 5, y 7.
Al grupo tres pertenecen los maxtérminos 8, 10, 12, 14, 24, 26, 28 y 30.
Al grupo cuatro pertenecen los maxtérminos 16, 18, 20, 22, 24, 26, 28 y 30.
El grupo uno queda como a' + b +d'.
El grupo dos queda como a + b + e'.
El grupo tres queda como b' + e.
El grupo cuatro queda como a' + e.

La función minimizada es:

$$F = (a' + b + d')\,(a + b + e')\,(b' + e)\,(a' + e)$$
$$F = a'\,b'\,e' + a\,d'\,e + b\,e$$

5.6 Utilidad de los términos opcionales

A veces, ciertas combinaciones de variables de entrada no ocurren nunca. Por ejemplo si se quiere diseñar un decodificador con base 12, habrá cuatro combinaciones que no se utilizarán. Para estas cuatro combinaciones no importa cuál sea la salida de la función, puesto que nunca ocurrirán. Estas condiciones que nunca ocurren pueden utilizarse para simplificar la función. Los términos opcionales se representan con una X, que significa "no importa que sea 1 ó 0".

5.7 Minimización con términos opcionales

Para minimizar utilizando términos opcionales, se siguen los mismos pasos que se utilizaron con anterioridad en los mapas. Las diferencias son: a) en el problema se indicará cuáles son los términos opcionales, y b) los términos opcionales se utilizarán únicamente cuando convengan para minimizar mejor la función.

Ejemplo. Diseñar un detector de múltiplos de 4, teniendo en cuenta para la minimización que a la entrada jamás se presentarán números impares. Resolverlo utilizando la suma de mintérminos.

Paso 1.- Primero se hará la tabla de verdad.

Entradas abcd	Salidas f
0000	0
0001	X
0010	0
0011	X
0100	1
0101	X
0110	0
0111	X
1000	1
1001	X
1010	0
1011	X
1100	1
1101	X
1110	0
1111	X

Como se quiere que el circuito nos indique un 1 con los múltiplos de cuatro, entonces se pone un 1 a la salida de cada combinación que representa un múltiplo de cuatro. Como se sabe que los números impares nunca se presentarán, se pone a la salida una X, sin importar que esas salidas sean 1 ó 0. Como los números pares sí se presentarán, entonces a la salida de los que no son múltiplos de cuatro se pondrá un 0. Por lo tanto, la función que se quiere simplificar será:

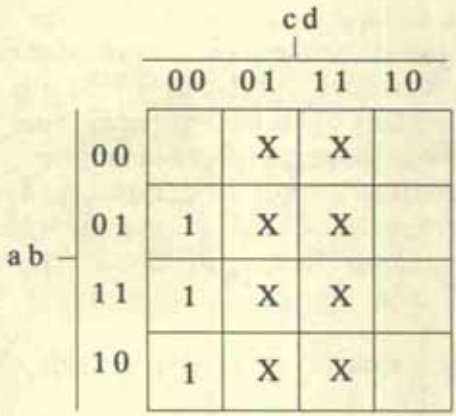
$$f(a, b, c, d) = \sum m\,(4, 8, 12) + d\,(1, 3, 5, 7, 9, 11, 13, 15)$$

donde

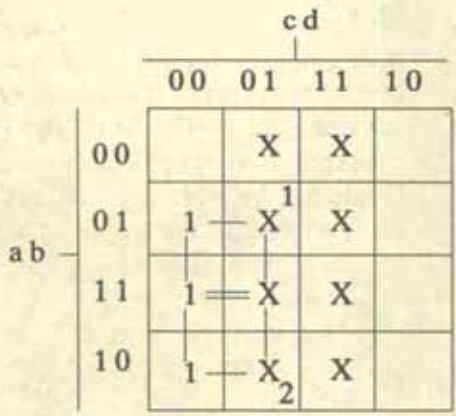
$$d\,(1, 3, 5, 7, 9, 11, 13, 15)$$

serán los términos opcionales.

Paso 2. Se localizan los mintérminos y los términos opcionales:



Paso 3. Se hace el agrupamiento:



Al grupo uno pertenecen los términos 4, 5, 12 y 13.
Al grupo dos pertenecen los términos 8, 9, 12 y 13.
El grupo uno queda como b c'.
El grupo dos queda como a c'.

La función minimizada es:

$$f = b\,c' + a\,c'$$

Si no se hubieran utilizado los términos opcionales, al minimizar la función hubiera quedado de dos términos de tres variables cada uno. Como se notará, sólo se utilizaron los términos opcionales que sirvieron; los otros no.

5.8 Conclusiones

Se aprendió a minimizar funciones en las que la minimización tiene mucho que ver con la habilidad y experiencia del diseñador. Si no se conoce bien el método se corre el riesgo de no llegar a la mejor minimización.

Al finalizar este tema se recomienda realizar la práctica número nueve.

5.9 Ejercicios

Realizar todos los problemas que se detallan a continuación, primero por suma de mintérminos y luego por producto de maxtérminos. En todos los circuitos dibuje el diagrama lógico sólo con compuertas NAND en el caso de mintérminos y sólo con compuertas NOR en el caso de maxtérminos.

1. Diseñar un decodificador de BCD a siete segmentos.
2. Diseñar un circuito de manera que permita comparar dos números X = X1 X2 y Y = Y1 Y2, y que dé un 1 a la salida cuando X sea menor o igual a Y.

3. Diseñar un circuito que detecte los números primos del 1 al 10. Supóngase que jamás se presentarán a la entrada del circuito los números no especificados.
4. Obtener el mínimo de componentes para la siguiente función:

$$F = w x' y z' + w' x y z' + w x' y' z + w' x y' z$$

5. Diseñar un decodificador de binario a código reflejado.

CAPÍTULO VI
Método tabular para la minimización de las funciones de Boole

La desventaja al utilizar los mapas de Karnaugh es que se trata de un procedimiento de ensayo y error. Se depende de la habilidad del diseñador. En funciones de seis o más variables es difícil tener la seguridad de que se hizo el mejor agrupamiento para la minimización. Si se examinan los ejemplos del capítulo anterior, en algunos se notará que puede haber soluciones alternativas; si no se domina el método de mapas quizá la solución que se encuentre no sea la mejor. Esto no sucede en el método de Quine y McCluskey, aunque éste último método es más laborioso que el de mapas de Karnaugh. Sin embargo, con un sencillo programa de computadora es posible superar esta complejidad.

6.1 Representación tabular

Este método fue formulado originalmente por Quine y fue mejorado posteriormente por McCluskey. Por esto se le da el nombre de los dos personajes. El método consta de dos pasos principales. Primero se encuentran los implicantes primos y en segundo lugar se localizan los implicantes primos esenciales.

6.2 Implicantes primos

Los implicantes primos, son un grupo de términos adyacentes candidatos para simplificar la función. Se trata de cualquier grupo de términos que no está incluido en un grupo mayor.

A continuación se verán algunos ejemplos de cómo minimizar funciones por este método. Primero se utilizará la notación binaria, para que quede clara la adyacencia de los términos. Después se utilizará la notación decimal, por ser más sencilla de trabajar.

Ejemplo 6.2.1. Minimizar la siguiente función utilizando el método de tabulación:

$$F = \sum m\,(1, 3, 4, 6, 9, 11, 12, 14)$$

Paso 1. Se agrupan los mintérminos de acuerdo con el número de unos que tienen en su representación binaria. Esto se muestra en las columnas (a) y (b) de la tabla 6.2.1.

Paso 2. Si dos términos difieren uno del otro sólo por una variable, podrán combinarse entre sí. La variable diferente se elimina y los términos que pueden combinarse se van marcando para asegurarse que ya están combinados. Se combinan los términos que tienen un uno con los que tienen dos unos y cuya diferencia entre sí es una potencia de dos. En seguida se combinan los que tienen dos unos con los que tienen tres unos, y así sucesivamente. Véase la columna (c) de la tabla 6.2.1.

Se separan por grupos. Se puede decir que el grupo uno en la columna (b) está formado por el 1 y el 4, y el grupo dos por el 3, el 6, el 9 y el 12. Se notará que el 4 y el 3 no se combinan por no haber adyacencia.

Paso 3. De nuevo se comparan los términos en cada sección, siempre y cuando tengan guion en la misma posición, pues eso significa que son adyacentes. En el ejemplo que se tiene, el término (00-1) se compara con el término (10-1). Por lo regular los equivalentes decimales se escriben en el lado izquierdo para facilitar la identificación. Aquí es relevante hacer notar que las comparaciones se van haciendo por grupos que están separados en cuadros. No se deben comparar grupos que no son adyacentes.

Pues bien, los términos no marcados son los implicantes primos que, en este caso, son los que definen los términos resultantes en la minimización. Se marcan con un asterisco.

Finalmente se encuentra y anota el resultado. Como los guiones indican que la variable se elimi-

(a)	(b)	(c)	(d)
número de unos	**un término**	**dos términos**	**cuatro términos**
	abcd	1,3 00-1 / 1,9 -001 /	1,3, 9,11 -0-0 *1 4,6,12,14 -1-0 *2
1	1 0001 / 4 0100 /	4,6 01-0 / 4,12 -100 /	
2	3 0011 / 6 0110 / 9 1001 / 12 1100 /	3,11 -011 / 6,14 -110 / 9,11 10-1 / 12,14 11-0 /	
3	11 1011 / 14 1110 /		

Tabla 6.2.1. Determinación de implicantes primos

na, en este caso se eliminan la a y la c, quedan en *1 b' d y en *2 b d'. Por lo tanto, el resultado es:

$$F = b\,d' + b'\,d$$

Se puede verificar el resultado que se haya encontrado si se resuelve el problema, ahora por el método de mapas. El resultado deberá ser el mismo.

El método tabular, sin embargo, es muy laborioso. Esto podrá simplificarse si en lugar de utilizar números binarios se utilizan números decimales. La única modificación que habrá que contemplar al utilizar números decimales es que éstos se restarán en lugar de compararse, como se hace con los números binarios. Cuando dos términos son iguales en todas las posiciones excepto en una, el término con el uno adicional debe ser mayor que el otro término por una potencia de dos. Para entenderlo mejor se resolverá de nuevo el ejemplo anterior ahora utilizando números decimales.

Ejemplo 6.2.2. Minimizar la siguiente función utilizando el método de tabulación y notación decimal.

$$F = \sum m\,(1, 3, 4, 6, 9, 11, 12, 14)$$

El dos y el ocho entre paréntesis indican que las variables que se van a eliminar son las que tienen peso binario de 8, en este caso la *a*, y la variable de peso binario dos, que sería la *c*.

(a)	(b)	(c)	(d)
número de unos	**un término**	**dos términos**	**cuatro términos**
		1,3 (2) / 1,9 (8) /	1,3, 9,11 (2,8) *1 4,6,12,14 (2,8) *2
1	1 / 4 /	4,6 (2) / 4,12 (8) /	
2	3 / 6 / 9 / 12 /	3,11 (8) / 6,14 (8) / 9,11 (4) / 12,14 (2) /	
3	11 / 14 /		

Tabla 6.2.2. Determinación de implicantes primos

En lo sucesivo sólo se utilizará la notación decimal.

6.3 Implicantes primos esenciales

Son aquellos que nos darán la expresión final óptima con el menor número de literales. Vale hacer la aclaración de que el ejemplo anterior fue escogido de manera que se obtuviera la expresión simplificada a partir de la suma de los implicantes primos. Sin embargo, en la mayoría de los casos la suma de implicantes primos no necesariamente forma la expresión con el número mínimo de términos. Esto se demuestra con el siguiente ejemplo. A la selección de implicantes primos se le llama implicantes primos esenciales, porque son fundamentales para la minimización.

Ejemplo 6.3.1. Minimice la siguiente función:

$$F(a, b, c, d) = \sum m(0, 2, 3, 6, 7, 8, 9, 10, 13)$$

Los grupos *1, *2, *3 y *4 son los implicantes primos. En seguida y mediante la siguiente tabla se obtendrán los implicantes primos esenciales.

(a)	(b)	(c)	(d)	
número de unos	**un término**	**dos términos**	**cuatro térm.**	
0	0 /	0,2 (2) /	0,2,8,10 (2,8)*3	a
		0,8 (8) /		
1	2 /		2,3,6, 7 (1,4)*4	b
	8 /	2,3 (1) /		
		2,6 (4) /		
2	3 /	2,10 (8) /		
	6 /	8,9 (1) *1	c	
	9 /	8,10 (2) /		
	10 /			
		3,11 (4) /		
3	7 /	6,14 (1) /		
	13 /	9,11 (4) *2	d	

Tabla 6.3.1. Determinación de implicantes primos

			Términos de la función								
G			0	2	3	6	7	8	9	10	13
R	a /	0, 2, 8, 10 (2, 8)	/	/				/		/	
U	b /	2, 3, 6, 7 (1, 4)		/	/	/	/				
P	c /	8, 9 (1)						/	/		
O	d /	9, 13 (4)							/		/
S			/	/	/	/	/	/	/	/	/

Tabla 6.3.2. Determinación de implicantes primos esenciales

Con la tabla se determina que los implicantes primos esenciales son:

$$b'd',\ a'd' \text{ y } ac'd = a + b + d$$

La tabla se llena entonces de la siguiente manera:

1. La primera hilera horizontal se llena con los mintérminos de la función.
2. La primera hilera vertical se llena con los grupos que se obtuvieron en la tabla de implicantes primos, poniendo primero los grupos que contienen más mintérminos y después los de menos. Se separan con una línea doble y cada grupo se etiqueta con una letra.

3. Se seleccionan los mintérminos en los que sólo hay una marca —el cero en este ejemplo. Se hace así porque este mintérmino sólo pertenece a este grupo. Se marca entonces este grupo pues se trata de un implicante primo esencial. En la última línea horizontal se marca el cero y todos los mintérminos que pertenecen al grupo a.

   El siguiente término que sólo tiene una marca es el tres. Se marca, lo mismo que todos los términos que están en el grupo al que pertenece el tres, y también se marca su grupo como un implicante primo esencial.

   Al final queda el trece con una sola marca. Se hace lo mismo de nuevo, y se marca el grupo al que pertenece como un implicante primo esencial. Se marca este término y también todos los que pertenecen a este grupo. El grupo C no quedó incluido porque los mintérminos que pertenecen a él ya están incluidos en los grupos a y d.

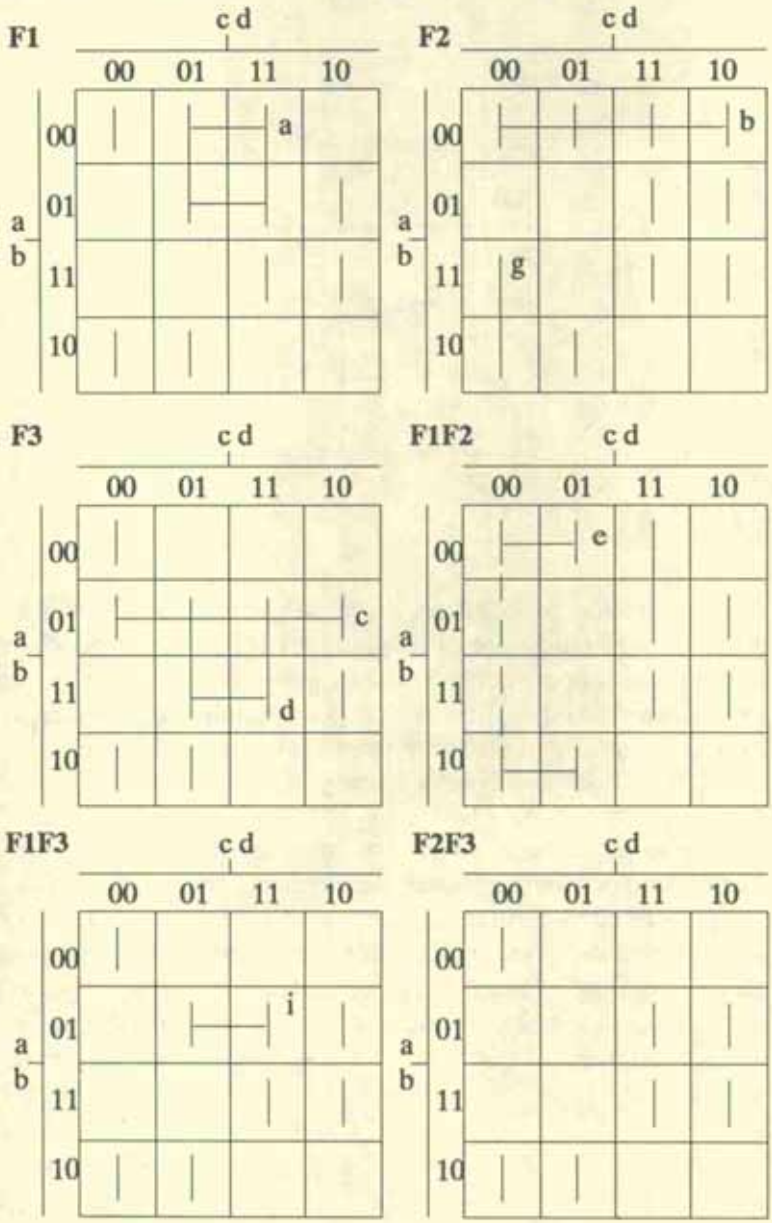
6.4 Circuitos de salida múltiple

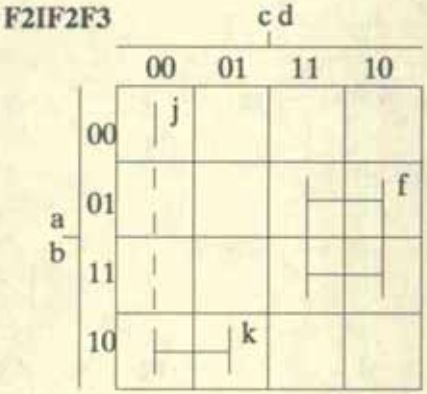
En los circuitos de salida múltiple se pueden combinar los métodos de mapas o utilizar únicamente el método tabular. A continuación se hará un ejemplo con la combinación de los dos métodos.

Ejemplo 6.4.1. Minimizar por medio de mapas el siguiente circuito de salida múltiple:

$$F1\ (a, b, c, d) = \sum m\ (0, 1, 3, 5, 6, 7, 8, 9, 14, 15)$$
$$F2\ (a, b, c, d) = \sum m\ (0, 1, 2, 3, 6, 7, 8, 9, 12, 14, 15)$$
$$F3\ (a, b, c, d) = \sum m\ (0, 4, 5, 6, 7, 8, 9, 13, 14, 15)$$

Primero se hace el mapa de F1, F2 y F3. En seguida se hace el mapa de las intersecciones de F1 F2, F1 F3, F2 F3 y F1 F2 F3.


F1
c d
00 01 11 10
a b
00
01
11
10
a
F2
b
g
F3
c
d
F1F2
e
F1F3
i
F2F3



A continuación se localizan los mintérminos y se forman los grupos, empezando por el mapa de F1 F2 F3. Los grupos ya seleccionados no se deberán volver a seleccionar a menos que formen parte de un grupo mayor, por ejemplo, el formado por los términos 6, 7, 14 y 15 como ya se seleccionaron en el mapa de F1 F2 F3, ya no se vuelven a seleccionar estos términos en los otros mapas.

Ya seleccionados los grupos se etiquetan con una letra, empezando por los grupos que tienen más términos. Los grupos que se obtienen no son el número mínimo. Para realizar la mejor de las minimizaciones se utiliza la tabla del método tabular para obtener los implicantes primos esenciales. Esta tabla se utilizará para las tres funciones, haciendo la especificación al lado izquierdo de cada grupo a qué función o a qué funciones pertenece. Ver tabla 6.4.1.

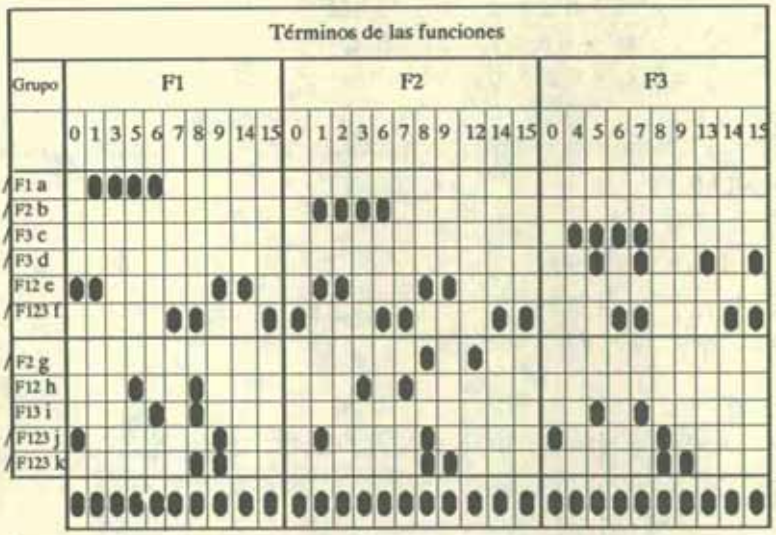


Tabla 6.4.1. Selección de implicantes primos esenciales para funciones de entradas y salidas múltiples

La tabla se llena de la misma manera que se hizo para una sola función. Para ello se observa la tabla cuidadosamente y se seleccionan los mintérminos que pertenecen a un solo grupo, hasta que todos los términos queden implicados. Sin embargo, no todos los grupos serán seleccionados. Los grupos e, i, y h no entran en la minimización, pues todos los términos que pertenecen a estos grupos ya están incluidos en otros grupos. La tabla indica que el resultado de la minimización es el siguiente:

F1 = grupo a + grupo f + grupo j + grupo k
F2 = grupo b + grupo f + grupo g + grupo j + grupo k
F3 = grupo c + grupo d + grupo f + grupo j + grupo k

Sustituyendo cada grupo por la función minimizada, nos queda:

$F1 = a'd + bc + b'c'd' + ab'c'$
$F2 = a'b' + bc + ac'd' + b'c'd' + ab'c'$
$F3 = a'b + bd + bc + b'c'd' + ab'c'$

Si se implementa el circuito quedaría como se muestra en la figura 6.4.1:

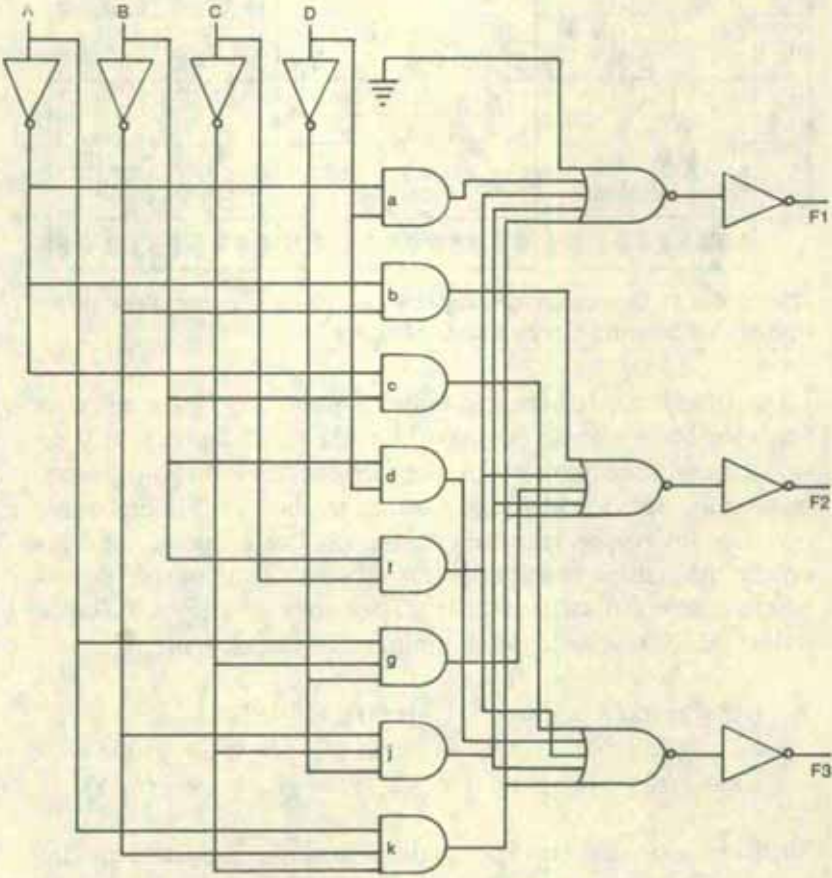


Figura 6.4.1. Circuito minimizado de entradas y salidas múltiples

6.5 Minimización de circuitos de salida múltiple

También se pueden determinar en forma tabular los implicantes primos de salida múltiple. Esto se demostrará con el mismo ejemplo anterior, donde ahora no se utilizarán mapas de Karnaugh.

*Ejemplo 6.5.*1. Determinar los implicantes primos del ejemplo anterior utilizando el método tabular y la notación decimal.

$$F1(a, b, c, d) = \sum m(0, 1, 3, 5, 6, 7, 8, 9, 14, 15)$$
$$F2(a, b, c, d) = \sum m(0, 1, 2, 3, 6, 7, 8, 9, 12, 14, 15)$$
$$F3(a, b, c, d) = \sum m(0, 4, 5, 6, 7, 8, 9, 13, 14, 15)$$

El procedimiento es igual que para una sola función. Se seleccionan los mintérminos por el número de unos. A cada mintérmino se le pondrá a un lado la función a la que pertenece y no se combinarán términos si no pertenecen a la misma función o funciones. Ver tabla 6.5.1.

Como se habrá notado, en la tabla se obtuvieron más grupos que los obtenidos con los mapas de Karnaugh. Sin embargo, éstos desaparecerán al hacer la selección de los implicantes primos esenciales. Esto es porque, por inspección, se pudo notar que estos grupos contenían mintérminos que ya antes habían sido seleccionados, o sea, que ya estaban contenidos en otros grupos más grandes.

Para seleccionar los implicantes primos esenciales se hace exactamente lo mismo que se hizo para seleccionarlos con los mapas de Karnaugh. Se deberá llegar al mismo resultado.

(a) (b) (c) (d)

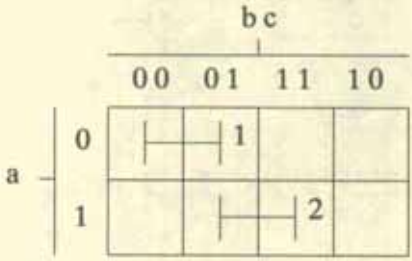
NÚM. DE UNOS	UN TÉRMINO	DOS TÉRMINOS	CUATRO TÉRMINOS
0	/0 [1, 2, 3]	/ 0, 1 [1, 2] (1) / 0, 2 [2] (2) 1 0, 4 [3] (4) j 0, 8 [1, 2, 3] (8)	b 0, 1, 2, 3 [2] (1, 2) e 0, 1, 8, 9 [1, 2] (1, 8)
1	/1 [1, 2] /2 [2] /4 [3] /8 [1, 2, 3]	m 1, 3 [1, 2] (2) / 1, 5 [1] (4) / 1, 9 [1,2] (8) / 2, 3 [2] (1) / 2, 6 [2] (4) / 4, 5 [3] (1) / 4, 6 [3] (2) k 8, 9 [1, 2, 3] (1) g 8, 12 [2] (4)	a 1, 3, 5, 7 [1] (2, 4) p 2, 3, 6, 7 [2] (1, 4) c 4, 5, 6, 7 [3] (1, 2)
2	/3 [1, 2] /5 [1, 3] /6 [1, 2, 3] /9 [1, 2, 3] /12 [2]	h 3, 7 [1, 2] (4) i 5, 7 [1, 3] (2) / 5, 13 [3] (8) / 6,7 [1, 2, 3] (1) / 6, 14 [1, 2, 3] (8) n 9, 13 [3] (4) o 12, 14 [2] (2)	d 5, 7, 13, 15[3] (2, 8) f 6, 7, 14, 15[1, 2, 3] (1, 8)
3	/7 [1, 2, 3] /13 [3] /14 [1, 2, 3]	/ 7, 15 [1, 2, 3] (8) /13, 15 [3] (2) /14, 15 [1, 2, 3] (1)	
4	/15 [1, 2, 3]		

Tabla 6.3.1. Determinación de implicantes primos

6.6 Riesgos en el diseño de circuitos combinacionales

En el capítulo III se vio la configuración interna de los circuitos integrados. Pudo notarse que los circuitos integrados no están hechos con dispositivos ideales. A estos dispositivos les toma tiempo conmutar de un estado a otro. Como ningún dispositivo es idéntico, la conmutación les toma más tiempo que a otros. Vale la pena tomar en consideración algunos aspectos, porque esto trae como consecuencia transiciones inesperadas de un estado a otro; o dicho en otras palabras, un nivel lógico inesperado al que se le puede llamar ruido. Este es un ejemplo de un circuito en cuyo diseño se pueden encontrar este tipo de anomalías.

Ejemplo 6.6.1. Diseñar el circuito formado por la suma de mintérminos $F = \Sigma m\,(0, 1, 5, 7)$. El mapa de Karnaugh para esta función será:



donde el grupo uno está formado por los mintérminos 0 y 1, y el grupo dos por los mintérminos 5 y 7. Por lo tanto, la función minimizada es:

$$F = a'\,b' + a\,c$$

Véase la figura 6.6.1. La salida de la compuerta AND #1 genera el término a' b' y la salida de la compuerta AND #2 genera el

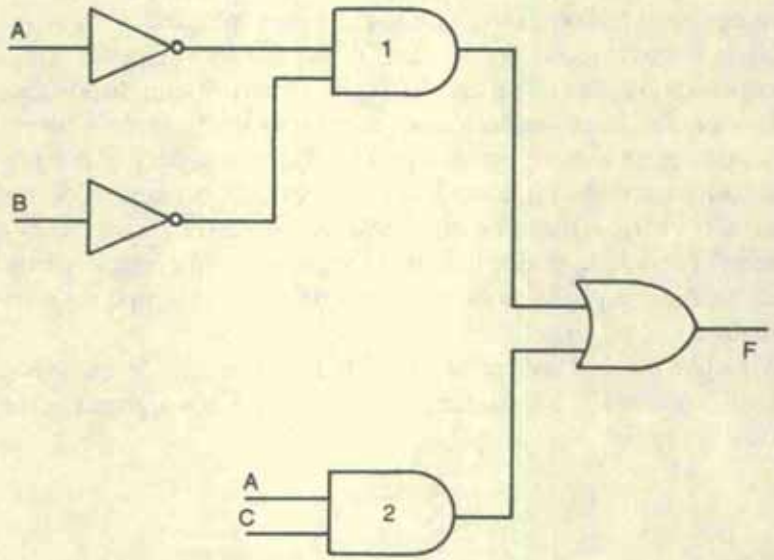


Figura 6.6.1. Circuito con riesgo estático

término a c. Supóngase que las entradas al circuito son ABC = 001, y que se cambia a de 0 a 1, para que ABC = 101. Primero se tiene que la salida de la compuerta #1 es 1, y la salida de la compuerta #2 es 0. Después del cambio en a se tendrá que la salida de la compuerta #1 es 0, y la de la compuerta #2 es 1. Sin embargo si la compuerta #1 cambia a 0 antes de que la compuerta número #2 cambie a 1, en un instante de tiempo se tendrá que la salida del circuito será 0. Si se determina que en el circuito no importa el valor que tenga la entrada a, cuando las entradas de

b c sean 0 1, la salida debe ser 1, y se presenta tal situación en la que transitoriamente la salida es 0. A esto es a lo que se llama un riesgo estático en los unos.

También puede haber circuitos con riesgos estáticos en los ceros. El remedio es añadir un grupo de mintérminos redundantes formado por 1 y 5, y por lo tanto, una compuerta redundante que genere un uno lógico, independientemente del valor de c. Esta nueva compuerta se muestra en la figura 6.6.2. El circuito queda ahora libre de riesgos.

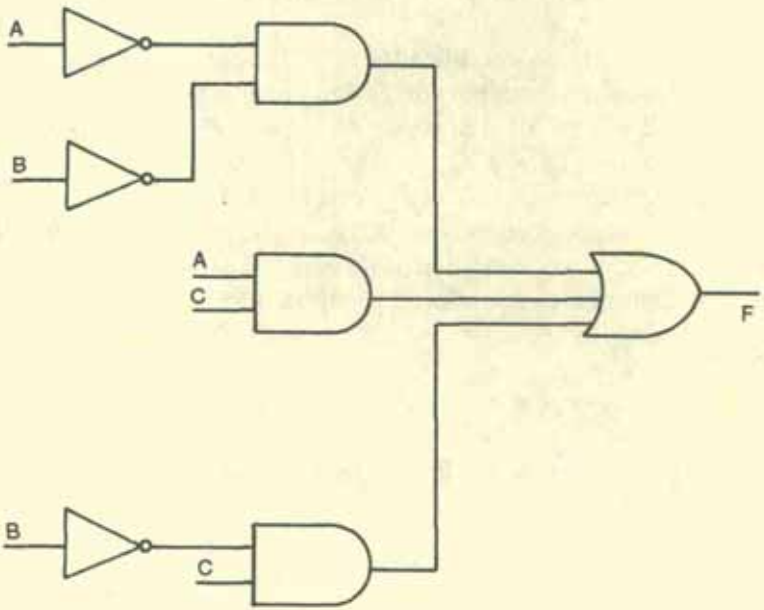


Figura 6.6.2. Circuito libre de riesgos

6.7 Conclusiones

Los capítulos dos, cinco y seis muestran los métodos más importantes de minimización. Son métodos para diseñar con circuitos combinacionales SSI que son las compuertas básicas. A continuación se analizarán circuitos combinacionales MSI. Estos métodos son útiles cuando se requiere diseñar un circuito con una aplicación muy especial no muy compleja.

6.8 Ejercicios

Resolver los siguientes problemas utilizando el método tabular:

1. Diseñar un decodificador de BCD a siete segmentos.
2. Diseñar un circuito que compare dos números X = X1 X2 y Y = Y1 Y2, y que dé 1 a la salida cuando X sea menor o igual a Y.
3. Diseñar un circuito que detecte los números primos del 1 al 10. Supóngase que jamás se presentarán a la entrada del circuito los números no especificados.
4. Obtenga el mínimo de componentes para la siguiente función:

$$F = w x' y z' + w' x y z' + w x' y' z + w' x y' z$$

5. Diseñar un decodificador de binario a código reflejado.

CAPÍTULO VII
Diseño con circuitos combinacionales

Antes de diseñar un sistema digital complejo con circuitos combinacionales es importante verificar si la función ya está disponible en algún circuito integrado. Hay muchos dispositivos que realizan funciones específicas para sistemas digitales de variados usos. Si no se encuentra el circuito quizá se pueda adaptar.

7.1 Tipos de circuitos integrados

Tanto en lógica combinacional como en lógica secuencial, los circuitos se clasifican por su complejidad de la siguiente manera:

SSI (*Small-Scale Integration*) Integración en pequeña escala.
MSI (*Medium-Scale Integration*) Integración en mediana escala.
LSI (*Large-Scale Integration*) Integración en gran escala.

Los circuitos SSI son los circuitos integrados de menos complejidad: compuertas lógicas y algunos arreglos que tienen hasta 12

compuertas en el circuito integrado. Este es el tipo de circuitos que se han visto en capítulos anteriores.

Los circuitos MSI son aquellos en los que un subsistema completo o una función de un sistema es fabricada o se obtiene a partir de un sólo circuito integrado. Un circuito MSI tiene más de 12 compuertas y menos de 100. Los circuitos LSI son sistemas mayores fabricados en un solo microcircuito. Tienen cien o más compuertas, o un circuito similar en complejidad. Los siguientes capítulos se referirán a los circuitos MSI y LSI más importantes, así como a algunas de sus aplicaciones.

7.2 Sumador

Con el método gráfico y el método tabular se aprendió a minimizar funciones obteniendo circuitos de 2 niveles. Ahora, si se quiere diseñar un sumador binario, la lógica a seguir será la siguiente:

$$\begin{array}{l} C_n, \ , \ , \ C_3 \ C_2 \\ a_n \ , \ , \ , \ a_3 \ a_2 \ a_1 + \\ b_n \ , \ , \ , \ b_3 \ b_2 \ b_1 = \\ \hline S_{n+1} = c_{n+1} \ \ S_n, \ , \ , \ S_3 \ S_2 \ S_1 \end{array}$$

Sería poco práctico intentar diseñar un circuito de dos niveles porque sería muy laborioso, ya que requeriría de un circuito de $n + 1$ salidas y $2n$ entradas. En una computadora, n puede ser igual a 32. Existe otro método para diseñar un circuito así.

Medio sumador

La operación aritmética básica es la suma de dos dígitos binarios:

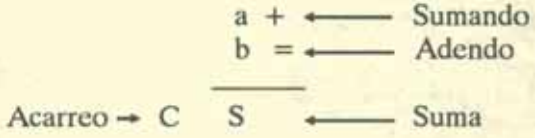
0 +	0 +	1 +	1 +
0 =	1 =	0 =	1 =
0	1	1	10

Las tres primeras operaciones producen una suma cuya longitud es un dígito. En la cuarta suma, cuando los dos bits son iguales a uno, la suma binaria consta de dos dígitos. El bit más significativo de este resultado se denomina acarreo. De aquí definiremos dos circuitos sumadores que son:

Medio sumador. Es un circuito combinacional que lleva a cabo la adición de dos bits.

Sumador completo. Es un circuito combinacional que lleva a cabo la adición de tres bits 2 bits significativos y una cuenta que se lleva de la etapa anterior.

Diseño de un medio sumador. Este circuito tendrá dos entradas y dos salidas binarias:



Se diseña la tabla de verdad para este circuito:

a	b	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

La función de Boole que se obtiene es la siguiente:

$$S = a'b + ab' \quad y \quad C = ab$$

El diagrama lógico y otras opciones se muestran en la figura 7.2.1.

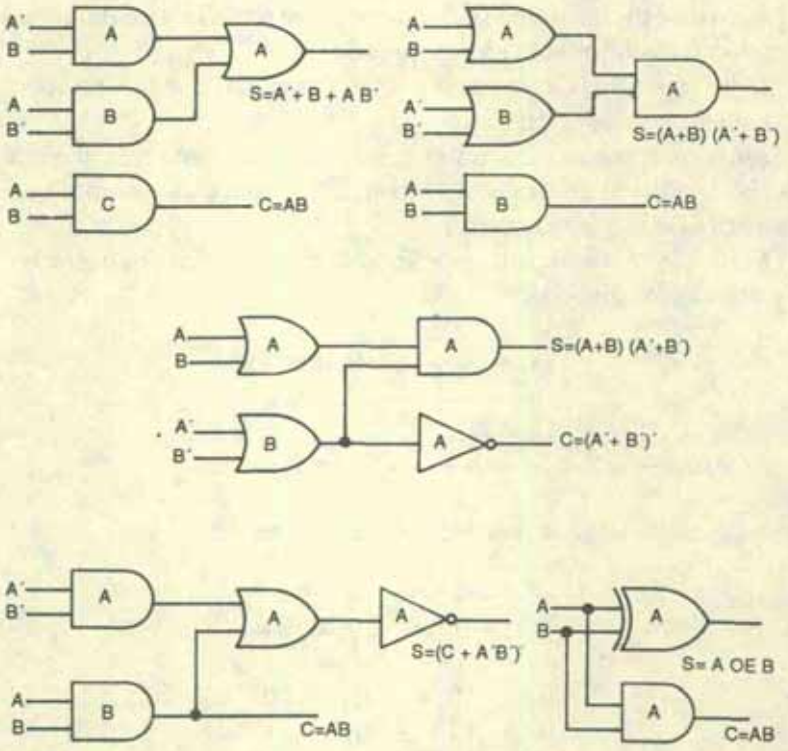


Figura 7.2.1. Diversas opciones para un medio sumador

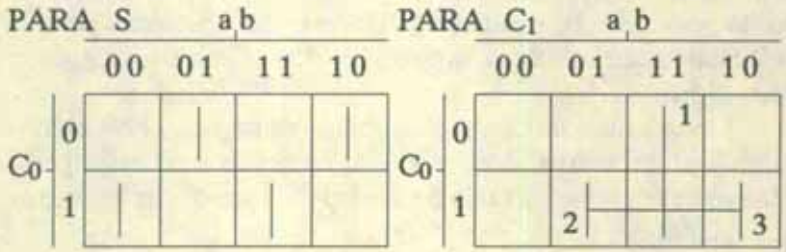
Diseño de un sumador completo. Como se dijo antes, consta de tres entradas y dos salidas. Las entradas serán a, b y C_0; a y b son los dos bits que van a sumarse. C0 representa la cuenta que se lleva de la posición previa menos significativa. Hay dos salidas porque el resultado de la suma varía de 0 a 3, y el 2 y el 3 binario requieren dos dígitos.

Las salidas son C_1 y S. S significa suma y es el bit menos significativo de la salida. C_1 significa *carry* y es el bit más significativo de la salida. La tabla de verdad del sumador completo es la siguiente:

C_0	a	b	C_1	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Tabla 7.2.1. Tabla de verdad de un sumador completo

Los mapas para un sumador completo serían los siguientes:



Como no se puede hacer algún grupo para S, queda como:

$$S = C_0' a' b + C_0' a b' + C_0 a' b' + C_0 a b$$

Para C_1 tendremos tres grupos:

El grupo 1 queda como a b.
El grupo 2 queda como C_0 b.
El grupo 3 queda como C_0 a.

Por lo tanto:

$$C_1 = a b + C_0 b + C_0 a$$

La implementación de este circuito se muestra en la figura 7.2.2.

Pueden desarrollarse otras configuraciones para un sumador completo. Por ejemplo, puede implementarse con dos medios sumadores y una compuerta OR. Para eso, obsérvense los mapas de aplicar el álgebra de Boole a S y C_1.

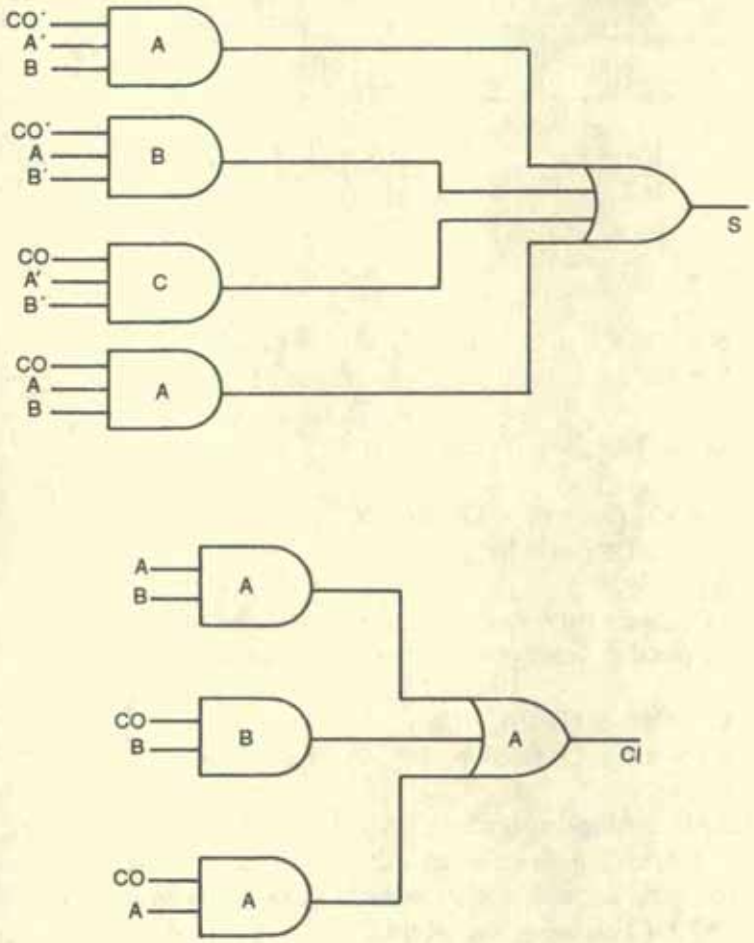
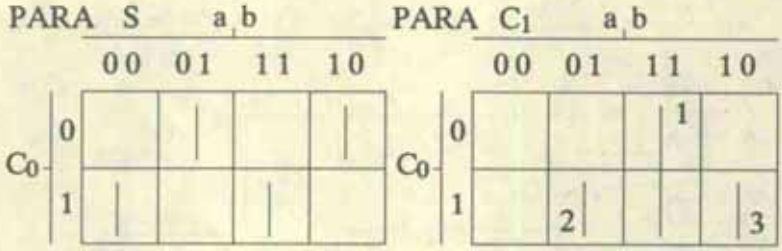


Figura 7.2.2. Implementación de un sumador completo en forma de suma de productos



Para S:

$$S = C_0' a' b + C_0' a b' + C_0 a' b' + C_0 a b$$
$$S = C_0' (a' b + a b') + C_0 (a' b' + a b)$$

(Nota: OE es igual a *OR exclusivo*)

$$S = C_0' (a \text{ OE } b) + C_0 (a \text{ OE } b)'$$
$$S = C_0 \text{ OE } (a \text{ OE } b)$$

Para C_1 se hacen de nuevo tres grupos diferentes, de manera que se puedan tener compuertas en común entre S y C_1:

$$C_1 = a b + C_0 a' b + C_0 a b'$$
$$C_1 = a b + C_0 (a \text{ OE } b) \qquad \text{(Ec. 7.2.1)}$$

La implementación para S y C_1 de esta configuración se muestra en la figura 7.2.3. Si se desea utilizar sólo compuertas NAND y OE se puede recurrir a la implementación que se muestra en la figura 7.2.4. Esto se obtiene al aplicar los teoremas de De Morgan a la Ec. 7.2.1:

$$C_1 = (a b + C_0 (a \text{ OE } b))''$$
$$C_1 = ((a b)' (C_0 (a \text{ OE } b))')'$$

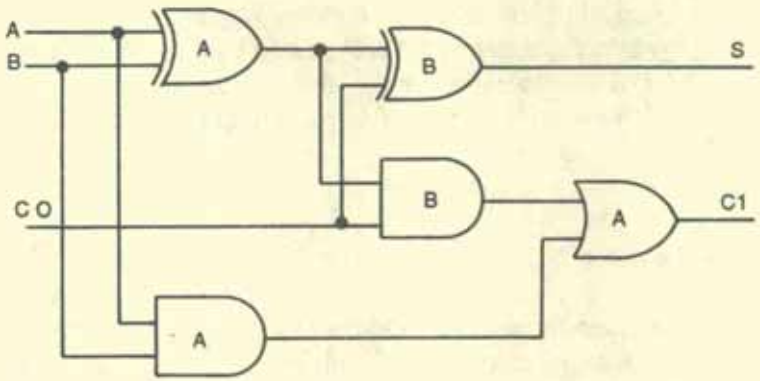


Figura 7.2.3. Sumador completo

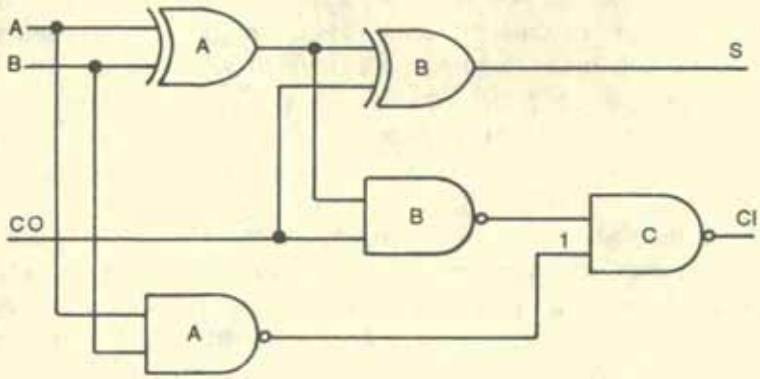


Figura 7.2.4. Otra opción para el sumador completo

7.3 Restador

La sustracción de dos números binarios puede llevarse a cabo tomando el complemento del sustraendo y agregándolo al minuendo. Por este método la operación de sustracción requiere de sumadores completos para su implementación en una máquina.

Medio restador

Un medio sumador es un circuito combinacional que sustrae dos bits y produce su diferencia. También tiene una salida para especificar si se ha tomado un 1 prestado. El bit del minuendo se designará con la letra *a* y el bit del sustraendo con la letra *b*. Para llevar a cabo a - b, tiene que verificarse la magnitud relativa de a y b. Si a es mayor o igual que b se tienen tres posibilidades: 0 - 0 = 0, 1 - 0 = 1 y 1 - 1 = 0. Al bit que resulta se le denomina diferencia. Si a < b, tenemos 0 - 1. Es necesario tomar un 1 de la siguiente etapa más alta. Este 1 que se toma añade un 2 al minuendo. El sistema es idéntico al que se utiliza en el sistema decimal sólo que en el sistema decimal se añade un 10 al minuendo en vez de un 2. Con el minuendo igual a 2, la diferencia será 2 - 1 = 1. Se deberá indicar que se pidió un 1 a la siguiente etapa más alta, si ésta no existe; entonces, el resultado es negativo.

El medio restador requiere de dos salidas. Una salida genera la diferencia y se denominará con la letra D. La otra salida indicará si se pidió prestado a la siguiente etapa más alta y se denominará con la letra P. La tabla de verdad del medio restador es la siguiente:

a	b	P	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Las funciones de Boole que definen a P y a D son:

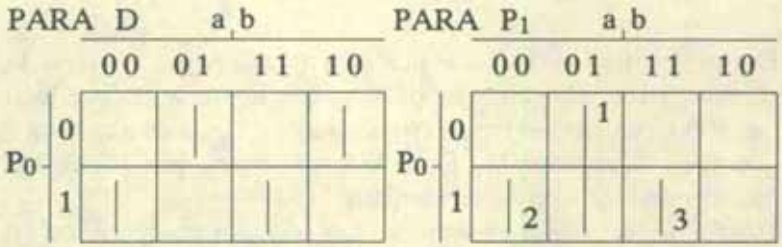
$$P = a'\,b$$
$$D = a'\,b + a\,b'$$

Es interesante observar que la lógica para D es exactamente la misma que la lógica para la salida S en el medio sumador. También que P es 0 cuando a es mayor o igual que b, y 1 cuando a es menor que b. Si se implementa el circuito se notará que difiere del medio sumador sólo en un inversor.

Diseño de un restador completo. Un restador completo es un circuito combinacional que lleva a cabo una sustracción entre dos bits, tomando en cuenta que un 1 se ha tomado por una etapa significativa más baja. Este circuito tiene tres entradas y dos salidas. Las entradas son a, b y P_0. Las salidas son D y P_1. La tabla de verdad para este circuito es la siguiente:

P_0	a	b	P_1	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

La función de Boole simplificada para las dos salidas del restador completo se ve en los siguientes mapas:



$$D = P_0 \text{ OE } (a \text{ OE } b)$$
$$P_1 = a' b + P_0 (a \text{ OE } b)'$$

La implementación de este circuito se muestra en la figura 7.3.1.

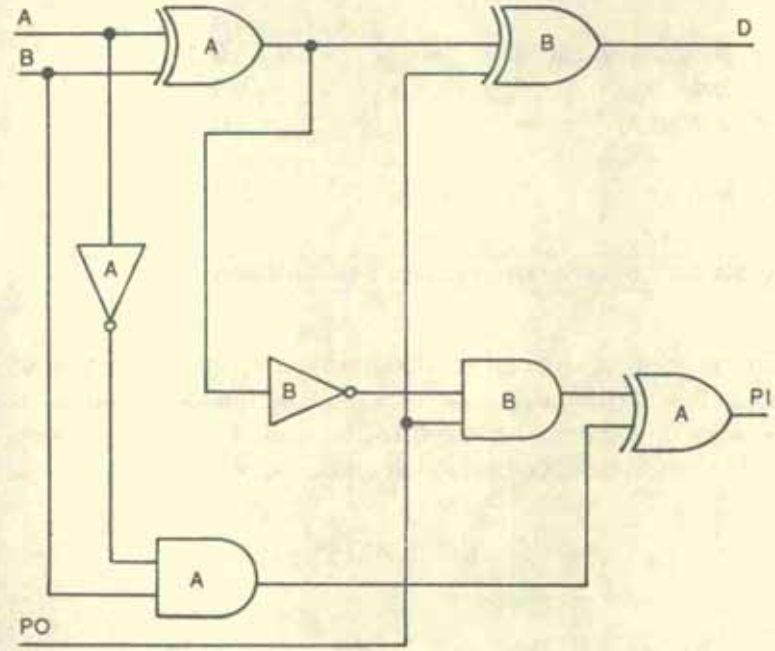


Figura 7.3.1. Restador completo

Sumador binario paralelo. El sumador completo que se vio forma la suma de dos bits y un previo acarreo. Dos números binarios de n bits cada uno pueden sumarse con un arreglo de n circuitos de estos. Esto puede quedar demostrado con un ejemplo específico. Considérense dos números binarios: a = 1011 y b = 0011, cuya suma es S = 1110. Cuando se agrega un par de bits a través de un sumador completo, el circuito produce un acarreo para usarse con el par de bits de una posición significativa más alta. Esto se muestra en la tabla 7.3.1:

Subíndice i			4 3 2 1
Acarreo de entrada		C_i	0 1 1 0
Sumando 1	+	a_i +	1 0 1 1 +
Sumando 2	=	b_i =	0 0 1 1 =
Suma		Si	1 1 1 0

Tabla 7.3.1. Suma de dos números de cuatro bits

En este caso, si se unen cuatro sumadores completos como se muestra en la figura 7.3.2, se construirá un sumador de números binarios en paralelo. La versión de un sumador en serie se verá en la sección de circuitos digitales secuenciales.

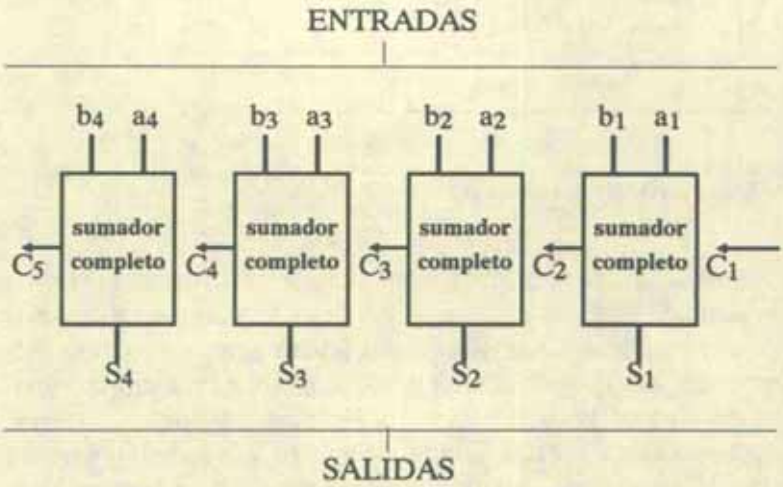


Figura 7.3.2.

Este es el primer circuito MSI combinacional que se ha visto. En los circuitos TTL éste tiene el número 74283 y se muestra en la figura 7.3.3:

Figura 7.3.3. Sumador binario

Propagación de acarreo. La suma de dos números binarios en paralelo implica que todos los bits del sumando y del aduendo estén disponibles para computarse al mismo tiempo. Si se considera la salida S_4 en un sumador, es posible darse cuenta de que las entradas b_4 y a_4 alcanzan un valor estacionario tan pronto como la señal de entrada se aplica al sumador. Sin embargo el acarreo de entrada C_4 no estará disponible hasta que C_3 envíe la señal. El retardo de la señal es una característica de los sistemas de nivel múltiple. C_3 también tiene que recibir la señal de C_2 y así sucesivamente. Así que no se tendrá el resultado correcto hasta que el acarreo se haya propagado completamente. El tiempo total de propagación desde C_1 a C_4 será de ocho niveles.

Una solución para reducir el tiempo de propagación de acarreo es utilizar compuertas rápidas. Otra técnica es la que se describe a continuación.

Considérese el circuito sumador completo que se muestra en la figura 7.3.4, donde se definen dos nuevas variables P_i y G_i, donde:

$$P_i = a_i \text{ OE } b_i \quad y \quad G_i = a_i \bullet b_i$$

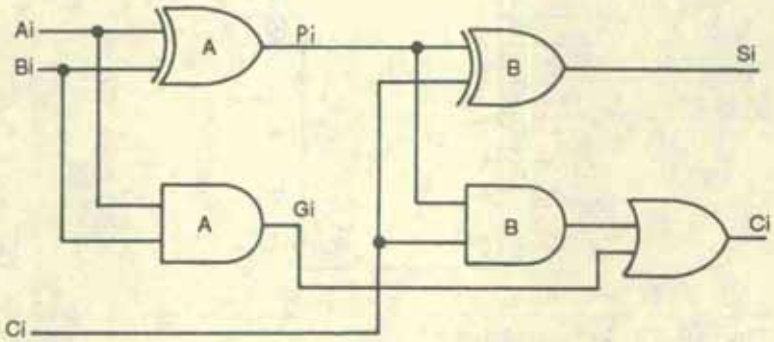


Figura 7.3.4. Sumador completo. Se definen dos nuevas variables

La suma y acarreo de salida pueden expresarse como:

$$S_i = P_i \text{ OE } C_i \quad y \quad C_{i+1} = P_i \bullet C_i + G_i$$

G_i es un acarreo generado y produce un acarreo de salida cuando tanto A_i como B_i son 1. P_i se denomina acarreo propagado. Ahora se aplicará la función de Boole para el acarreo de salida en cada etapa y se sustituye su valor en cada Ci mediante la ecuación previa.

$C_2=P_1 \bullet C_1+G_1$
$C_3=P_2 \bullet C_2+G_2=P_2 \bullet P \bullet C_1+P_2 \bullet G_1+ G_2$
$C_4=P_3 \bullet C_3+G_3=P_3 \bullet P_2 \bullet P_1 \bullet C_1+P_3 \bullet P_2 \bullet G_1+P_3 \bullet G_2+G_3$

Ya que la función de Boole para cada acarreo de salida se expresa en suma de productos, cada función puede implementarse con un nivel de compuertas AND seguido por una compuerta OR. Las tres funciones de Boole para C_2, C_3 y C_4 se implementan en el generador de acarreo por anticipado que se muestra en la figura 7.3.5. Obsérvese que C_4 no tiene que esperar para que se propaguen C_2 y C_3. De hecho, C_4 se propaga al mismo tiempo que C_2 y C_3. Un generador de acarreo por anticipado es el 74182. Se implementa con compuertas AND-OR-inversoras. El IC 74182 se muestra en la figura 7.3.6. La aplicación típica de un generador de acarreo por anticipado se muestra en la figura 7.3.7.

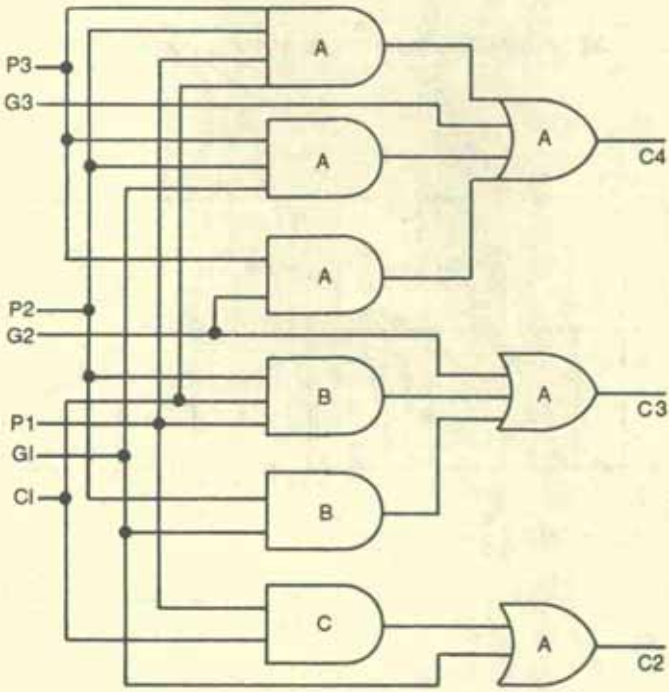


Figura 7.3.5. Diagrama lógico de un generador de acarreo por anticipado

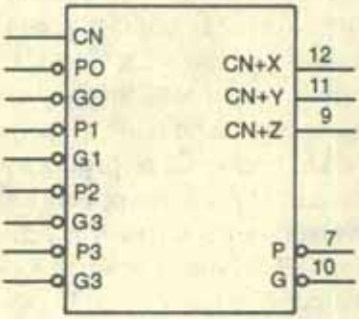


Figura 7.3.6. IC 74182. Generador de acarreo por anticipado

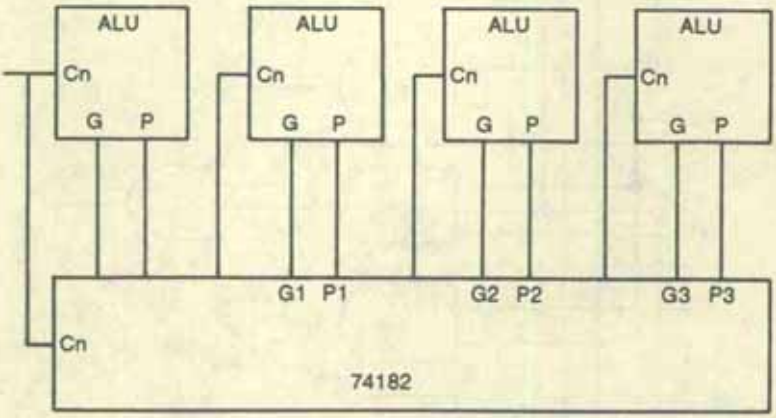


Figura 7.3.7. Aplicación típica de un generador de acarreo por anticipado

La construcción de un sumador paralelo de cuatro bits con un esquema de acarreo por anticipado se muestra en la figura 7.3.8. Ahí los acarreos se propagan a través del generador de acarreo por anticipado y se aplican como entradas a la segunda compuerta OE. Así S_2, S_3 y S_4 tienen tiempos iguales de propagación.

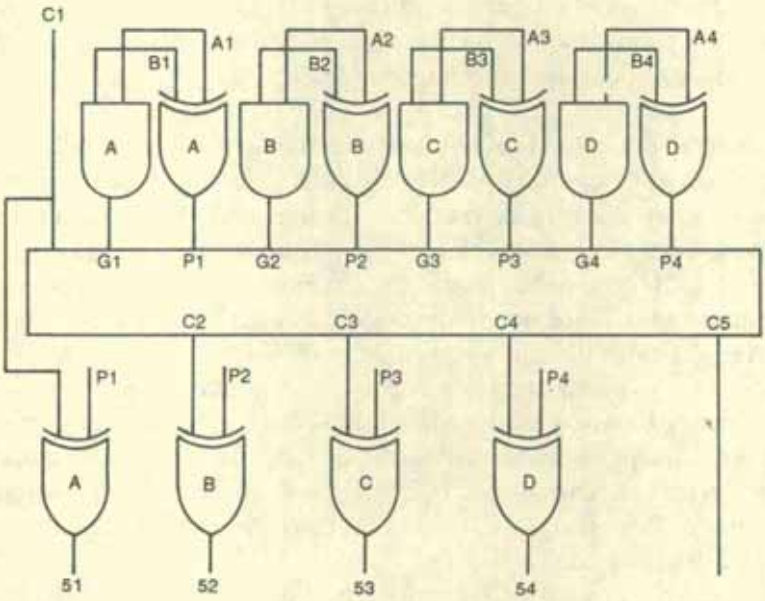


Figura 7.3.8. Sumador paralelo de cuatro bits con acarreo por anticipado

7.4 Conversión de código

Los sistemas digitales utilizan diferentes códigos. A veces, la salida de un sistema es la entrada a otro sistema. Si cada uno de

los sistemas utiliza un código diferente para la misma información, debe insertarse un circuito de conversión entre los sistemas. Un convertidor de código es un circuito que hace dos sistemas compatibles aun cuando cada uno use un código binario diferente. Para convertir un código binario A en el código binario B, las líneas de entrada deben suministrar el código A al convertidor y las salidas deben generar el código B. La transformación se lleva a cabo con compuertas lógicas. Antes conviene conocer algunos códigos binarios importantes.

Códigos binarios. Los sistemas electrónicos digitales utilizan señales que tienen dos valores distintos y elementos de circuitos que tienen dos estados estables. Hay una analogía directa entre las señales binarias, los elementos del circuito binario y el dígito binario. Un número binario de n dígitos, por ejemplo, puede representarse por n elementos de números binarios, cada uno con una señal de salida equivalente a 0 o a 1.

Los sistemas digitales representan y manipulan no sólo números binarios, también muchos otros elementos de información. Cualquier elemento discreto de información distinto entre un grupo de cantidades puede representarse por un código binario. Por ejemplo, el rojo es un color definido del espectro. La letra A es una letra del alfabeto.

Un bit, es un dígito binario. Un grupo de cuatro cantidades diferentes puede representarse mediante un código de 2 bits. Un grupo de ocho elementos requiere un código de 3 bits. Hay grupos en los que su número de elementos no es una potencia de dos y las combinaciones de bits que restan quedan sin asignarse. Por ejemplo, tenemos el grupo de los diez dígitos decimales 0, 1, ...9. Un código binario para diez elementos debe tener cuando menos cuatro bits. Las seis combinaciones restantes simplemente se quedan sin asignar y no se utilizan.

Aunque el número mínimo de bits necesarios para codificar 2_n cantidades distintas es n, no hay un número máximo de bits

establecido para representar un código binario. Por ejemplo, los diez dígitos decimales pueden codificarse con diez bits.

Dígito decimal	Código binario decimal de diez bits
0	0000000001
1	0000000010
2	0000000100
3	0000001000
4	0000010000
5	0000100000
6	0001000000
7	0010000000
8	0100000000
9	1000000000

Códigos decimales. Los códigos binarios para dígitos decimales requieren un mínimo de cuatro bits. Se obtienen numerosos códigos al combinar cuatro o más bits. Unas cuantas posibilidades son mostradas en la siguiente tabla:

Dígito Decimal	BCD	Exceso-3	84-2-1	2421	Biquinario 5043210
0	0000	0011	0000	0000	0100001
1	0001	0100	0111	0001	0100010
2	0010	0101	0110	0010	0100100
3	0011	0110	0101	0011	0101000
4	0100	0111	0100	0100	0110000
5	0101	1000	1011	1011	1000001
6	0110	1001	1010	1100	1000010
7	0111	1010	1001	1101	1000100
8	1000	1011	1000	1110	1001000
9	1001	1100	1111	1111	1010000

Estos códigos se logran asignando pesos a los bits binarios de acuerdo con sus posiciones. También se les pueden asignar valores negativos a un código decimal como se ve en el código 84 -2 -1.

De los cinco códigos binarios que se enlistan en la tabla, el BCD es el más común. Los otros códigos de cuatro bits listados tienen una característica común que no se encuentra en el BCD. El código de exceso a 3, el 2421, y el 84 -2 -1 son códigos autocomplementarios; esto es que el complemento a 9 del número decimal se obtiene fácilmente cambiando los unos a ceros y los ceros a unos. Por ejemplo, el decimal 395 se representa en el código 2421 por 0011 1111 1011 y su complemento a nueve, que es el 604 será el 110000000100.

El código biquinario de siete bits tiene propiedades de detección de error. Cada dígito decimal consta de cinco números 0 y dos números 1. Al transmitir una señal puede ocurrir un error. El receptor puede detectar el error cuando al recibir la señal ésta tiene más de dos unos.

Códigos de detección de error. Al transmitir una señal cualquier ruido externo que se introduzca cambia los valores de bits de 0 a 1 y viceversa. Puede utilizarse un código para detectar el error. El error no se corrige pero se detecta. Uno de los métodos que se utilizan es agregar un bit de paridad para hacer que el número total de unos sea par o impar. Ejemplo:

Al ser posible determinar que los datos recibidos de una línea de transmisión o muestreados de una memoria son erróneos, se podría efectuar una retransmisión o un remuestreo de datos. El método más simple se conoce con el nombre de verificación de paridad. Supóngase, por ejemplo, que la información se debe almacenar en una cinta magnética con caracteres de siete bits binarios cada uno. Entonces se añade un octavo bit a cada caracter, de manera que el número de bits 1 del caracter sea siempre par. Luego el caracter se codifica con paridad par.

MENSAJE	P(impar)	P(par)
0000	1	0
0001	0	1
0010	0	1
0011	1	0
0100	0	1
0101	1	0
0110	1	0
0111	0	1
1000	0	1
1001	1	0
1010	1	0
1011	0	1
1100	1	0
1101	0	1
1110	0	1
1111	1	0

Al llegar la información se verifica que el número de bits 1 siga siendo par. Por ejemplo, establézcase la paridad par adicionando un bit a cada uno de los caracteres de siete bits.

CARÁCTER DE 7 BITS	PARIDAD PAR (8 BITS)
1101001	11010010
0101111	01011111

La determinación es rápida. La función de Boole para comprobar la paridad impar es un ejemplo clásico de una función que no se realiza eficientemente en dos niveles.

	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

El mapa de Karnaugh sería como un tablero de ajedrez por sus múltiples posibilidades. Para verificar la paridad de ocho variables en un circuito de dos niveles se requerirían 1,129 compuertas NAND.

El circuito que se utiliza para detectar la paridad impar de dos bits es el circuito OE. El circuito que verifica la paridad impar de cuatro variables se muestra en la figura 7.4.1. El circuito nos dará

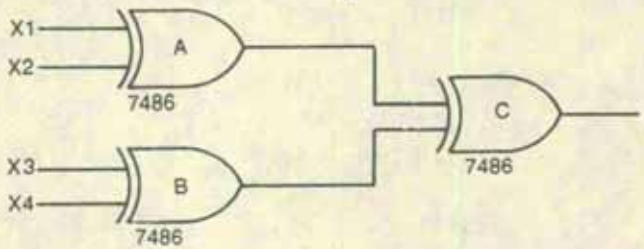


Figura 7.4.1. Verificador de paridad impar de cuatro variables

un 1 cuando el número de unos sea impar. La figura 7.4.2 muestra un verificador de paridad impar de ocho variables.

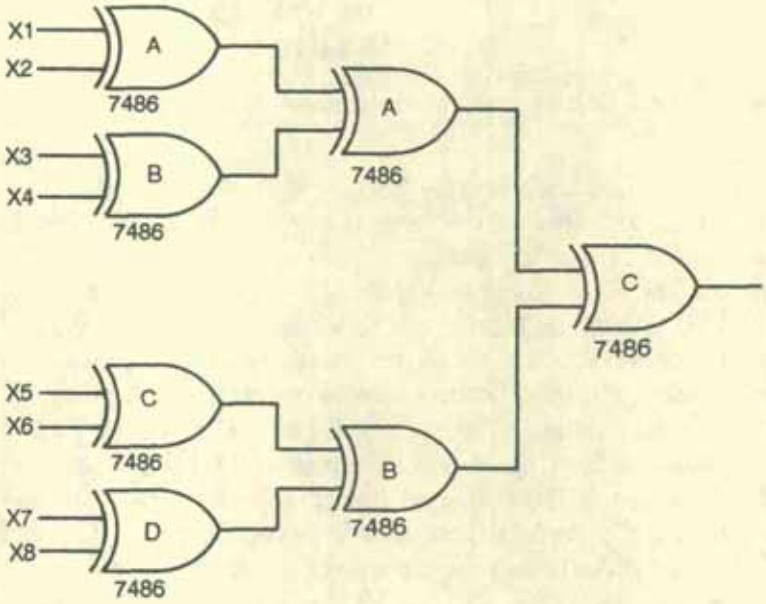


Figura 7.4.2. Verificador de paridad impar de ocho variables

Al observar la reducción de probabilidad de error que se deriva al añadir un solo bit de paridad, se puede pensar en estudiar la posibilidad de agregar más de un bit redundante. Este método llevará a una reducción subsecuente en la probabilidad de error y, como se demostrará, puede incluso facilitar la corrección sin necesidad de retransmisión.

Supóngase que los cuatro caracteres codificados de la siguiente figura son los únicos que se transmitirán mediante un sistema de comunicaciones:

A	00000
B	11100
C	00111
D	11011

Figura 7.4.3. Código para transmitirse con M=3

Obsérvese que cada uno de los cuatro caracteres codificados difiere de los otros en por lo menos tres de cinco bits. Por lo tanto se dice que la distancia mínima del código es tres. En general, la distancia mínima M de un código se define como el número mínimo de bits en que difieren dos caracteres cualesquiera del código. En los cuatro caracteres anteriores existen cuatro caracteres codificados específicos de dos bits. Por lo tanto, la distancia mínima M = 3 se ha logrado al agregar tres bits redundantes. Supóngase que se transmite D, pero la señal se recibe como 1 1 0 0 0. Los dos últimos bits se recibieron equivocados pero no se confundirán ni con A, ni con B, ni con C. En efecto, dos bits equivocados en cualquier caracter no harán que se confunda con otro. Así, los errores en dos o menos bits se pueden detectar en cualquier código con una distancia mínima de tres. Los errores en tres o más bits no se van a poder detectar.

Si al transmitir el código de la figura se supone que no habrá más de un bit equivocado entonces se puede corregir el error. Por ejemplo si al transmitir 1 1 1 0 0 se recibe 1 1 0 0 0, si se supone que sólo hay un bit equivocado se detecta que el tercer bit es el que está mal. Se podrá corregir el error de un bit sin necesidad de retransmisión. Supóngase que se produce un error de dos bits en un caracter de un código de distancia mínima tres, establecido para corregir errores de un solo bit. El proceso de corrección se efectuará pero el caracter codificado seguirá equivocado. Por tanto, el error de dos bits no se detectará. Esto obedece a la siguiente ecuación:

$$M - 1 = C + D,$$

donde C es el número de bits equivocados y D es el número de errores que se detectan. C debe ser menor o igual que D.

Código de Hamming. Es un código de distancia mínima tres. En este código, las posiciones de los bits se numeran en secuencia de izquierda a derecha. Las posiciones numeradas como una potencia de 2 se reservan para los bits de verificación de paridad. Los restantes son bits de información. En seguida se presenta el código de siete bits:

1	2	3	4	5	6	7
P_1	P_2	X_3	P_4	X_5	X_6	X_7

A partir de este código se determinan los bits de paridad:

- P_1 establece la paridad par sobre 1 3 5 7
- P_2 establece la paridad par sobre 2 3 6 7
- P_4 establece la paridad par sobre 4 5 6 7

Ejemplo. Determinar el caracter codificado de acuerdo con Hamming, que corresponde al caracter de información:

$$X_3\ X_5\ X_6\ X_7 = 1\ 0\ 1\ 0$$

Se deberá establecer el valor de P1, P2 y P4:

$P_1 = 1$ para que P_1 OE X_3 OE X_5 OE $X_7 = 0$
$P_2 = 0$ para que P_2 OE X_3 OE X_6 OE $X_7 = 0$
$P_4 = 1$ para que P_4 OE X_5 OE X_6 OE $X_7 = 0$

Por lo tanto, el caracter codificado de acuerdo con el código de Hamming que se deberá transmitir será 1 0 1 1 0 1 0.

El proceso de corrección en el extremo receptor corrige si un bit está equivocado. Esto se hace de la siguiente manera: C_n deberá ser 0, a menos que haya algún error en la transmisión; entonces será 1. Por lo tanto:

$$C_1 = P_1 \text{ OE } X_3 \text{ OE } X_5 \text{ OE } X_7$$
$$C_2 = P_2 \text{ OE } X_3 \text{ OE } X_6 \text{ OE } X_7$$
$$C_4 = P_4 \text{ OE } X_5 \text{ OE } X_6 \text{ OE } X_7$$

Si hay algún error en algún bit del caracter, este se determinará mediante la siguiente tabla:

	Bit con error							
	*	1	2	3	4	5	6	7
C_4 (4, 5, 6, 7)	0	0	0	0	1	1	1	1
C_2 (2, 3, 6, 7)	0	0	1	1	0	0	1	1
C_1 (1, 3, 5, 7)	0	1	0	1	0	1	0	1

* En este caso no existe error alguno en la transmisión.

Ejemplo 1. Supóngase que se recibe el caracter:

$$C_1\, C_2\, X_3\, C_4\, X_5\, X_6\, X_7 = 1\,1\,0\,1\,1\,0\,1$$

Detectar si la información transmitida contiene o no errores. Si los hay, determinar cuál es el bit equivocado.

$C_1 = P_1$ OE X_3 OE X_5 OE X_7
$C_1 = 1$ OE 0 OE 1 OE 1 = 1
$C_2 = P_2$ OE X_3 OE X_6 OE X_7
$C_2 = 1$ OE 0 OE 0 OE 1 = 0
$C_4 = P_4$ OE X_5 OE X_6 OE X_7
$C_4 = 1$ OE 1 OE 0 OE 1 = 1

El bit equivocado es el $X_5 = 1$. Este debe ser igual a 0. El caracter correcto es 1 1 0 1 0 0 1.

Ejemplo 2. Codificar el caracter de información 0 1 1 0 1 1 1 0 1 0 1 de acuerdo con el código de Hamming de quince bits.

C_1	C_2	X_3	C_4	X_5	X_6	X_7	C_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
	0		1	1	0		1	1	1	0	1	0	1	

Se deberán determinar P_1, P_2, P_4 y P_8. Esto se hace de la siguiente manera:

$C_1 = P_1$ OE X_3 OE X_5 OE X_7 OE X_9 OE X_{11} OE X_{13} OE X_{15}
$0 = P_1$ OE 0 OE 1 OE 0 OE 1 OE 1 OE 1 OE 1
$C_2 = P_2$ OE X_3 OE X_6 OE X_7 OE X_{10} OE X_{11} OE X_{14} OE X_{15}
$0 = P_2$ OE 0 OE 1 OE 0 OE 1 OE 1 OE 0 OE 1
$C_4 = P_4$ OE X_5 OE X_6 OE X_7 OE X_{12} OE X_{13} OE X_{14} OE X_{15}
$0 = P_4$ OE 1 OE 1 OE 0 OE 0 OE 1 OE 0 OE 1
$C_8 = P_8$ OE X_9 OE X_{10} OE X_{11} OE X_{12} OE X_{13} OE X_{14} OE X_{15}
$0 = P_8$ OE 1 OE 1 OE 1 OE 0 OE 1 OE 0 OE 1

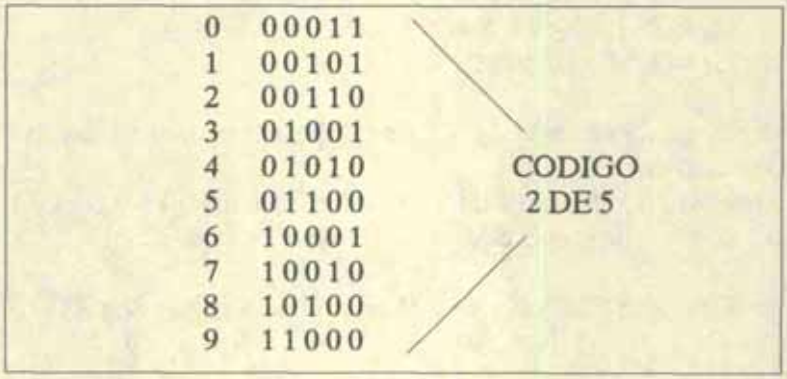
Se encuentra que:

$$P_1 = 1, P_2 = 0, P_4 = 0 \text{ y } P_8 = 1.$$

Por tanto, el caracter de información que se transmitirá será:

1 0 0 0 1 1 0 1 1 1 1 0 1 0 1

Otro código es el 2 de 5. Se caracteriza porque tiene dos unos por cada cinco bits. Este código se muestra a continuación:



Códigos alfanuméricos. Muchas de las aplicaciones de las computadoras digitales requieren de la manipulación de datos, que constan no sólo de números sino también de letras. Para representar un nombre es necesario tener un código binario para el alfabeto. El mismo código debe representar números decimales, letras y algunos otros caracteres especiales, como $. El número total de elementos en un grupo alfanumérico es mayor a 36; por lo tanto, debe codificarse con un mínimo de seis bits. El código ASCII es ejemplo de un código alfanumérico, aunque no es el único. Es un código de siete bits, por lo regular se hace de ocho bits. El octavo bit que se agrega es de paridad. En seguida se verán algunos ejemplos donde se ven aplicaciones de lo estudiado en este capítulo:

Problema 1. Convertir el código BCD en el código de exceso-3. Ambos códigos se muestran a continuación.

Entrada BCD				Salida código Exceso-3			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

Por inspección Z = D'. Con los mapas de Karnaugh se encontrarán W, X y Y.

PARA W

a b \ c d	00	01	11	10
00				
01	1	1	1	
11	x	x	x	x
10	1	1	x	x

PARA X

a b \ c d	00	01	11	10
00		1	1	1
01	1			
11	x	x	x	x
10		1	x	x

PARA Y

a b \ c d	00	01	11	10
00	1		1	
01	1		1	
11	x	x	x	x
10	1		x	x

Finalmente se determina que:

W = a + b d + b c
X = b c'd' + b'd + b'c
Y = c'd' + c d
Z = d'

El diagrama lógico de dos niveles se obtiene a partir de los mapas de Karnaugh, aunque hay también otras posibilidades derivadas también de los mapas. Una de ellas se muestra a continuación:

$$W = a + b\,(c + d)$$
$$X = b\,(c + d)' + b'\,(c + d)$$
$$Y = (c + d)' + c\,d$$
$$Z = d'$$

El diagrama lógico se muestra en la figura 7.4.4.

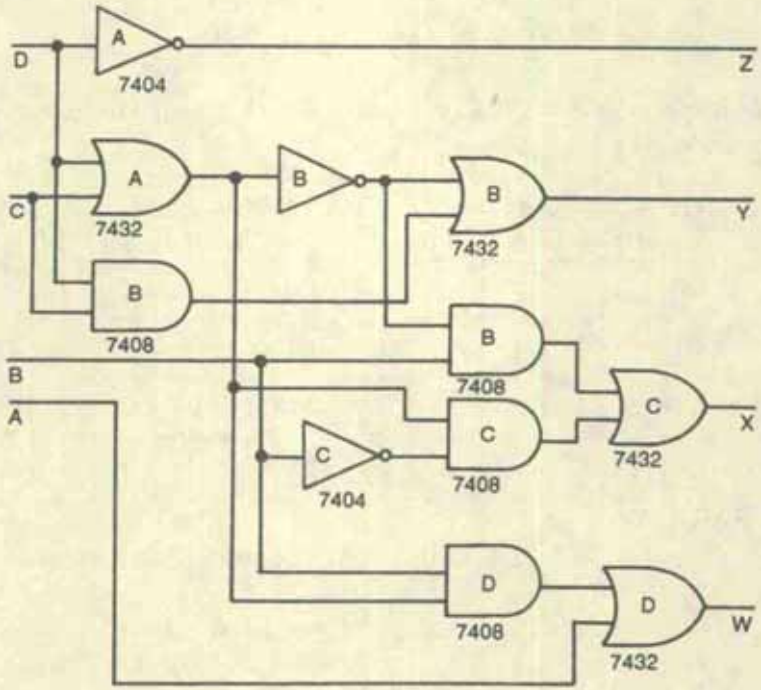


Figura 7.4.4. Convertidor de código BCD a Exceso-3 utilizando el método clásico

Con lo que se ha aprendido hasta ahora se está preparado para diseñar y analizar circuitos digitales combinacionales. Es necesario definir ambos términos.

Diseño. Aquí se empieza con especificaciones verbales de una función requerida y culmina con un diagrama lógico.

Análisis. Es el proceso inverso del diseño. Se empieza con un diagrama lógico y se termina con un conjunto de funciones de Boole, una tabla de verdad y una explicación verbal de la operación del circuito.

Para analizar un circuito combinacional es recomendable tener la seguridad de que es combinacional y no secuencial. El diagrama de un circuito combinacional tiene compuertas lógicas sin trayectorias de retroalimentación o elementos de memoria. Una trayectoria de retroalimentación es una conexión de la salida de una compuerta a la entrada de una segunda compuerta que forma parte de la primera compuerta.

Problema 2. Diseñar un convertidor de código de exceso-3 a BCD utilizando el circuito MSI de la figura 7.3.3.

El código BCD se obtiene sumándole al código exceso-3 el número binario 1101. Este número es el complemento de 0011. Esto implica que se le resta al código de exceso-3 el número tres, obteniendo así el código BCD. Ejemplo:

Número	+ Entrada exceso-3	= Salida BCD
1101	+ 1000	= 0101

Nota.- El acarreo no se utiliza.

La implementación del circuito del problema 2 se muestra en la figura 7.4.5. El dígito exceso-3 se aplica a las entradas a. Las entradas b se establecen en un 1101 constante. Este diseño es mucho más económico que el diseño por el método clásico del problema 1.

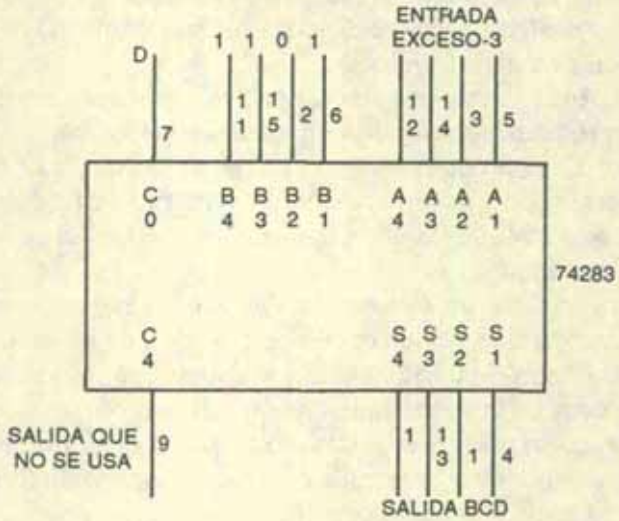


Figura 7.4.5. Convertidor de código de exceso-3 a binario utilizando un sumador

Sumador decimal. Las computadoras o calculadoras que realizan operaciones aritméticas directamente en el sistema de números decimales representan números decimales en forma de código binario. Un sumador debe emplear circuitos que acepten números decimales codificados y den resultados en códigos aceptados. Un sumador decimal requiere un mínimo de nueve entradas y cinco salidas, ya que se necesitan cuatro bits para codificar cada dígito decimal y el circuito debe tener un acarreo de entrada y un acarreo de salida.

Sumador BCD. Considérese la adición aritmética de dos dígitos decimales en código BCD, junto con un posible acarreo de una etapa anterior. Ya que cada dígito de entrada no excede de nueve, la suma de salida no puede ser mayor que 19: 9+9+1=19,

donde el 1 en la suma es un acarreo de entrada. Supóngase que se aplican dos dígitos BCD a un sumador binario de cuatro bits. El sumador formará la suma en binario y produce un resultado que puede variar de 0 a 19. Estos números se listan en la siguiente tabla y se etiquetan con los símbolos K, Z_8, Z_4, Z_2 y Z_1. K es el acarreo y los subíndices de la letra Z representan el peso binario de cada dígito 8, 4, 2 y 1 que se asigna a los cuatro bits del código BCD. El problema será encontrar una regla simple por la que pueda convertirse el número binario de la primera columna en la representación del dígito correcto BCD del número en la segunda columna que se muestra en la siguiente tabla:

SUMA BINARIA					SUMADOR BCD					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

Al examinar el contenido de la tabla, es evidente que cuando la suma binaria es igual o menor que 1001, el número correspondiente BCD es idéntico y, por tanto, no se necesita conversión. Cuando la suma binaria es mayor que 1001, se obtiene una representación BCD que no es válida. La adición del binario 0110 a la suma binaria la convierte en la representación BCD correcta y también produce un acarreo de salida cuando se requiere. El circuito lógico que detecta la corrección necesaria puede derivarse mediante las entradas de la tabla. Es obvio que se requiere una conexión cuando la suma binaria tiene un acarreo de salida K = 1. Las otras seis combinaciones de 1010 a 1111 que necesitan una corrección tienen un 1 en la posición Z_8. Se especifica además que Z_4 o Z_2 deben tener un 1. La condición para corrección y acarreo de salida pueden expresarse por la función de Boole:

$$C = K + Z_8 Z_4 + Z_8 Z_2$$

Cuando C = 1 es necesario agregar 0110 a la suma binaria y proporcionar un acarreo de salida para la siguiente etapa. Un sumador BCD es un circuito que suma dos dígitos BCD en paralelo y produce una suma digital también en BCD. Un sumador BCD debe de incluir la lógica de corrección en su construcción interna. Para sumar 0110 se usa un segundo sumador binario de cuatro bits. Los dos dígitos decimales, junto con el acarreo de entrada, se suman primero en el sumador binario de cuatro bits para producir la suma binaria. Cuando el acarreo de salida es igual a cero, no se agrega cosa alguna en la suma binaria; cuando es igual a uno se añade el binario 0110 a la suma binaria a través del sumador binario de cuatro bits de abajo que se muestra en la figura 7.4.6. El acarreo de salida generado por el sumador binario de abajo se ignora, pues éste se suministra en la terminal de acarreo del sumador de arriba.

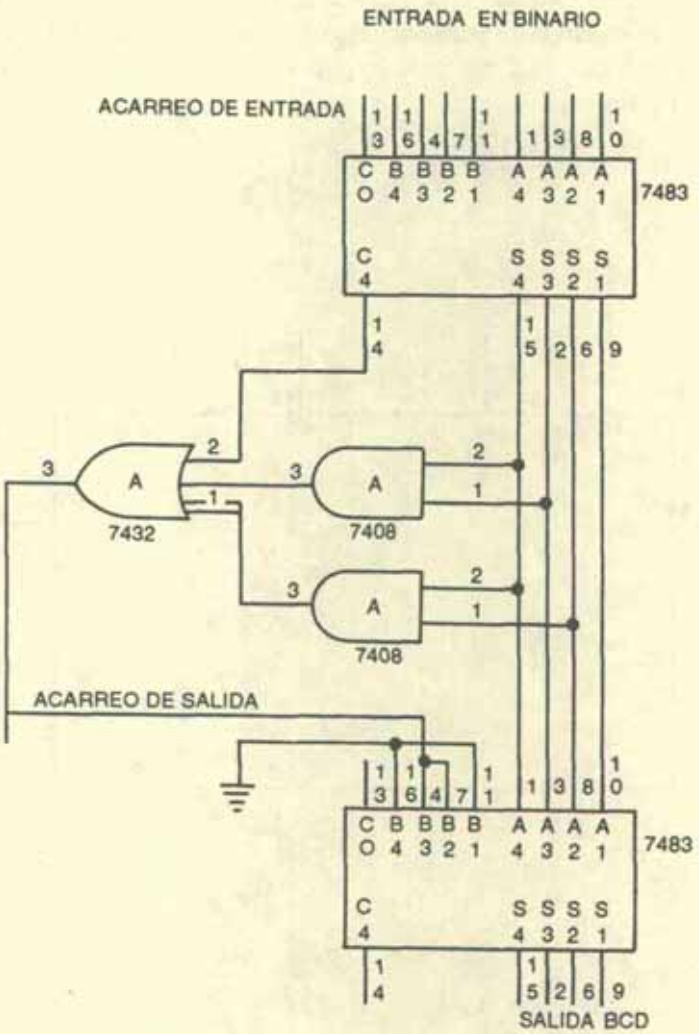


Figura 7.4.6. Diagrama de bloques de un sumador BCD

El sumador BCD puede construirse con tres ICs. Cada uno de los sumadores de 4-bits es una función MSI, y las tres compuertas para la corrección lógica necesitan un paquete SSI. El TTL 74F583 es un sumador BCD en un solo paquete. Este se muestra en la figura 7.4.7.

Figura 7.4.7. IC 74F583. Sumador de números decimales

7.5 Codificadores

Un codificador realiza la operación inversa de un decodificador. Un codificador tiene 2^n o menos líneas de entrada, y n líneas de salida generan el código binario para las 2^n variables de entrada. Un ejemplo de un codificador se muestra en la figura 7.5.1. El IC 74148 y el IC 74147 son codificadores que ya vienen en un IC cada uno.

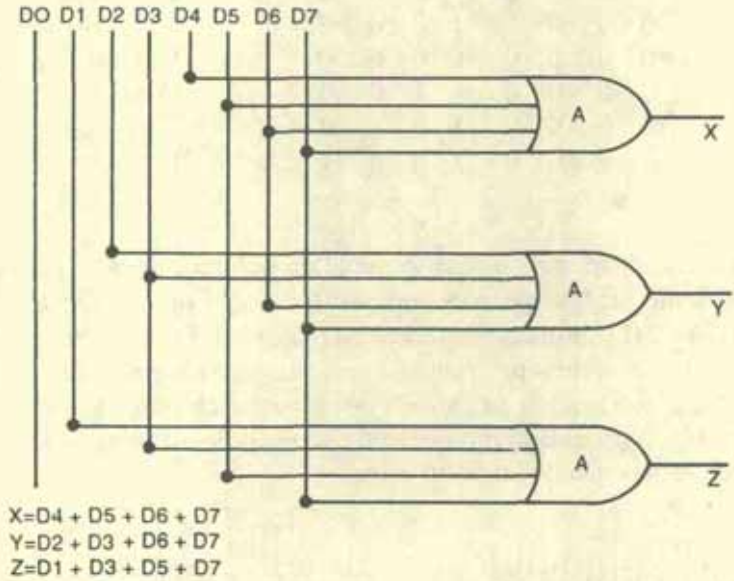


Figura 7.5.1 Codificador de octal a binario

El codificador de octal a binario consta de ocho entradas, una para cada uno de los ocho dígitos, y tres salidas que generan el número binario correspondiente. Se construye con compuertas OR. Su tabla de verdad es la siguiente:

Tabla de verdad de un decodificador de octal-a-binario										
ENTRADAS								SALIDAS		
D0	D1	D2	D3	D4	D5	D6	D7	X	Y	Z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Este codificador no está disponible en paquete pues se puede construir fácilmente con compuertas OR. Los codificadores 74147 y 74148 son codificadores de prioridad. Estos codificadores establecen una prioridad de entrada para asegurar que sólo se codifique la línea de entrada de más alta prioridad. Prioridad significa aquí que al momento de tener dos entradas se le da preferencia a la de más alto valor.

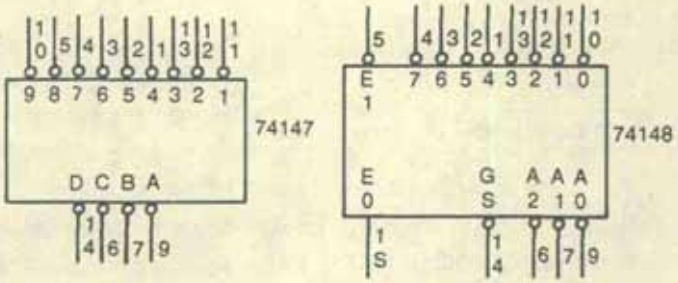


Figura 7.5.2 Se muestra el IC 74147 que es un codificador de decimal a binario. El IC 74148 es un codificador de octal a binario.

7.6 Decodificadores

Las cantidades discretas de información se representan en sistemas digitales con códigos binarios. Un código binario de n bits es capaz de representar hasta 2^n elementos distintos de información codificada. Un decodificador es un circuito combinacional que convierte información binaria de n líneas de entrada a un máximo de 2^n líneas únicas de salida. Si la información decodificada de n bits tiene combinaciones no usadas o sin importancia, la salida del decodificador tendrá menos de 2^n salidas.

Los decodificadores que aquí se presentan se llaman decodificadores n-a-m líneas, donde m es menor o igual que 2^n. Su propósito es generar 2^n o menos mintérminos de las n variables de entrada. El nombre decodificador también se utiliza junto con algunos convertidores de código como un decodificador BCD a 7 segmentos. Como ejemplo véase el circuito de la figura 7.6.1. Este es un decodificador de 3-8 líneas. Una aplicación de este decodificador sería la de utilizarlo como convertidor de binario a octal. Las variables de entrada representarán al número binario y la salida los ocho dígitos del sistema octal. El 74138 es un decodificador de 3-a-8 líneas. Está construido con compuertas NAND. Las salidas son los complementos de los valores que se muestran en la siguiente tabla de verdad. Las variables de salida son mutuamente excluyentes debido a que sólo una salida puede ser igual a uno.

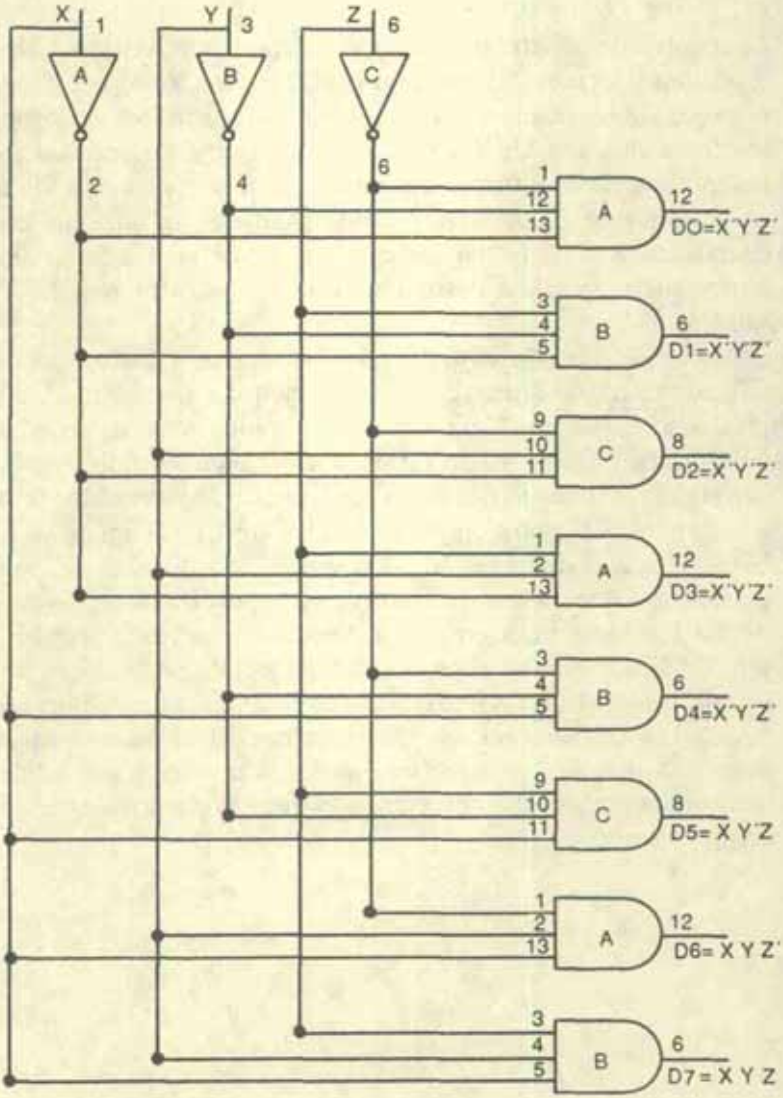


Figura 7.6.1 Decodificador de 3-8 líneas

Tabla de verdad de un decodificador de 3-a-8 líneas

Entradas			Salidas							
X	Y	Z	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Ejemplo 7.6.1. Diseñe un decodificador BCD-a-decimal. El decodificador debe tener cuatro entradas para aceptar el dígito codificado y diez salidas, una para cada dígito decimal. No se necesita diseñarlo pues ya existe en el mercado con el nombre IC 7442, pero es un buen ejemplo para demostrar las consecuencias prácticas de las condiciones *no importa*.

Si se diseña por el método clásico se necesitaría dibujar diez mapas para simplificar cada una de las funciones. Se deben tomar en cuenta las 6 condiciones no importa para simplificar. En lugar de dibujar diez mapas, se dibuja sólo uno y se escribe cada una de las variables de salida, D0 a D9 en el interior del cuadro de su mintérmino correspondiente, como se muestra en el siguiente mapa:

a b \ c d	00	01	11	10
00	D0	D1	D3	D2
01	D4	D5	D7	D6
11	X	X	X	X
10	D8	D9	X	X

Las condiciones *no importa* se usarán para simplificar la función al número mínimo de literales:

D0 = a' b' c' d'
D1 = a' b' c' d
D2 = b' c d'
D3 = b' c d
D4 = b c' d'
D5 = b c' d
D6 = b c d'
D7 = b c d
D8 = a d'
D9 = a d

El circuito se muestra en la figura 7.6.2. Los términos no importa causan una reducción en el número de entradas en la mayoría de las compuertas AND.

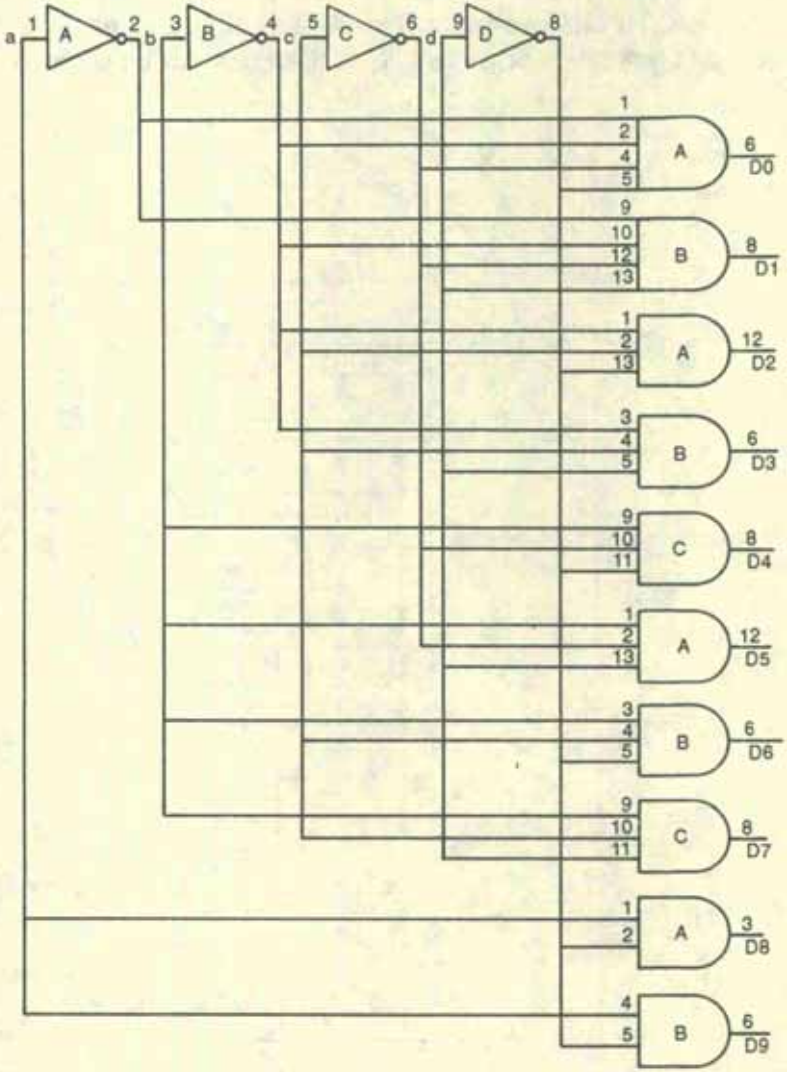


Figura 7.6.2. Decodificador de BCD a decimal utilizando las condiciones *no importa*

El IC 7442 es un decodificador de BCD-a-decimal. Las salidas seleccionadas están en estado 0 y todas las combinaciones inválidas dan una salida de todos 1. Este IC se muestra en la figura 7.6.3.

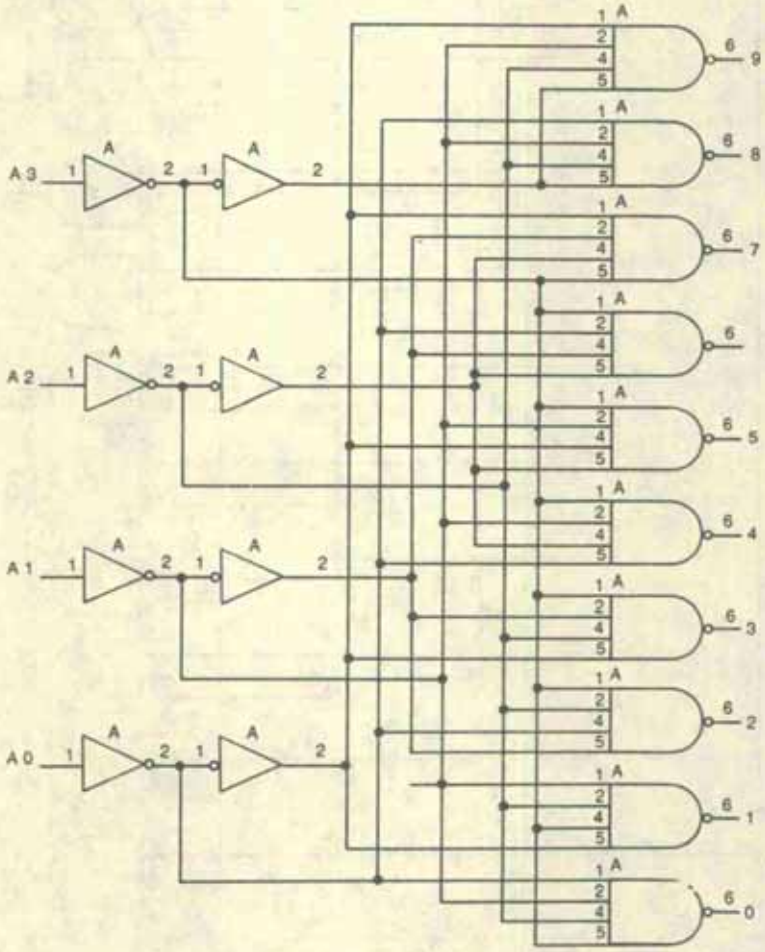


Figura 7.6.3. Decodificador BCD con cuatro entradas y diez salidas

Hay que tener cuidado cuando se diseña con condiciones no importa pues se supone que éstas no ocurrirán nunca. ¿Pero qué tal si se daña el circuito de la etapa anterior y envía datos erróneos? La siguiente tabla muestra qué pasaría si se tienen algunas de las condiciones no importa en la entrada. Si se quiere minimizar el número de errores, dependiendo de la exactitud requerida en el circuito, lo más conveniente es diseñar un decodificador de 10 compuertas AND de 4 entradas.

Tabla de errores que se podrían presentar al diseñar un circuito utilizando condiciones *no importa*

Entradas	Salidas
a b c d	D0 D1 D2 D3 D4 D5 D6 D7 D8 D9
1 0 1 0	0 0 1 0 0 0 0 0 1 0
1 0 1 1	0 0 0 1 0 0 0 0 0 1
1 1 0 0	0 0 0 0 1 0 0 0 1 0
1 1 0 1	0 0 0 0 0 1 0 0 0 1
1 1 1 0	0 0 0 0 0 0 1 0 1 0
1 1 1 1	0 0 0 0 0 0 0 1 0 1

Aplicaciones. Un decodificador proporciona 2^n mintérminos de n variables de entrada. Las funciones de Boole se expresan como suma de mintérminos. Por tanto, se puede utilizar un decodificador para generar esos mintérminos y una compuerta OR para formar la suma de mintérminos. Esto quiere decir que cualquier circuito combinacional con n entradas y m salidas puede implementarse con un decodificador de n a 2^n y m compuertas OR. Lo único que hay que hacer es elegir un decodificador que genere los mintérminos que se necesitan.

Ejemplo 7.6.2. Diseñar un restador completo con un decodificador y dos compuertas OR.

Un restador completo tiene dos salidas. Su tabla de verdad se muestra en seguida:

P_0	A	B	P_1	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

De su tabla de verdad se concluye que estas dos salidas se expresan mediante la siguiente suma de mintérminos:

$$P_1 (P_0, A, B) = \Sigma\, m(1, 4, 5, 7)$$
$$D (P_0, A, B) = \Sigma\, m(1, 2, 4, 7)$$

Puesto que el circuito tiene tres entradas, se necesita un decodificador de 3-a-8 líneas. Esto da por resultado el circuito de la figura 7.6.4. En este circuito el decodificador generó los ocho mintérminos. La salida P se derivó de la suma de los mintérminos 1, 4, 5 y 7, y la D de la suma de los mintérminos 1, 2, 4 y 7. Cada una de estas salidas se obtiene con una compuerta OR. Si se tiene una función con una lista larga de mintérminos se requerirá una compuerta OR con un gran número de entradas. Una función F que tenga una lista de k mintérminos puede expresarse en su forma complementaria F' con 2^n - k mintérminos. Si el número

de mintérminos en una función es mayor de $2^n/2$, entonces F' se puede expresar con menos mintérminos que los requeridos para F. Por tanto, se usará una compuerta NOR para la suma de mintérminos de F'. Esta generará la salida normal de F.

El método del codificador puede utilizarse para implementar cualquier circuito combinacional. Sin embargo, esta opción de diseño debe compararse con otras para determinar la mejor solución.

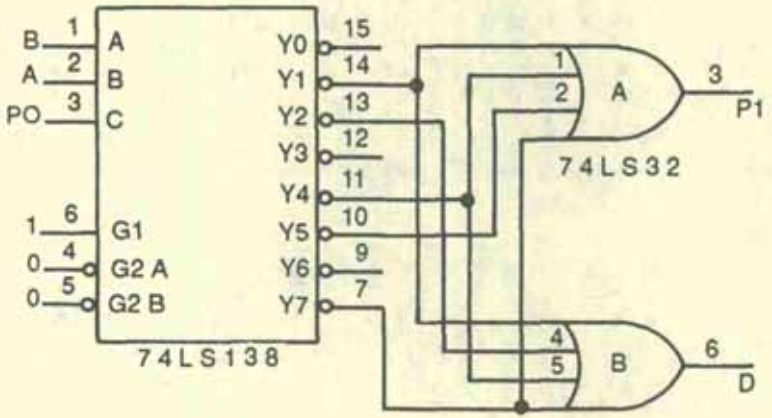
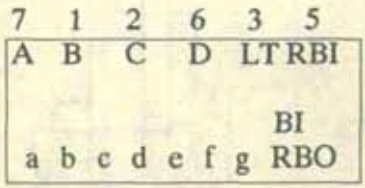


Figura 7.6.4. Restador completo obtenido con un decodificador

	ENTRADAS						SALIDAS							
FUNCIÓN DECIMAL	LT	RBI	D	C	B	A	BI/RBO	$\overline{a}$	$\overline{b}$	$\overline{c}$	$\overline{d}$	$\overline{e}$	$\overline{f}$	$\overline{g}$	NOTA
0	H	L	L	L	L	L	H	L	L	L	L	L	L	H	A
1	H	X	L	L	L	H	H	H	L	L	H	H	H	H	A
2	H	X	L	L	H	L	H	L	L	H	L	L	H	L	
3	H	X	L	L	H	H	H	L	L	L	L	H	H	L	
4	H	X	L	H	L	L	H	H	L	L	H	H	L	L	
5	H	X	L	H	L	H	H	L	H	L	L	H	L	L	
6	H	X	L	H	H	L	H	H	H	L	L	L	L	L	
7	H	X	L	H	H	H	H	L	L	L	H	H	H	H	
8	H	X	L	L	L	L	H	L	L	L	L	L	L	L	
9	H	X	H	L	L	H	H	L	L	L	H	H	L	L	
10	H	X	H	L	H	L	H	H	H	H	L	L	H	L	
11	H	X	H	L	H	H	H	H	H	L	L	H	H	L	
12	H	X	H	H	L	L	H	H	L	H	H	H	L	L	
13	H	X	H	H	L	H	H	L	H	H	L	H	L	L	
14	H	X	H	H	H	L	H	H	H	H	L	L	L	L	
15	H	X	H	H	H	H	H	H	H	H	H	H	H	H	
BI	X	X	X	X	X	X	L	H	H	H	H	H	H	H	B
RBI	H	L	L	L	L	L	L	H	H	H	H	H	H	H	C
LT	L	X	X	X	X	X	H	L	L	L	L	L	L	L	D

H = nivel alto de voltaje L = nivel bajo de voltaje X = no importa



Vcc = Pin 16
Gnd = Pin 8

SN54/74LS47

7.7 Comparador de magnitud

La comparación de dos números es una operación que determina si un número es mayor que, menor que o igual a otro número. Un comparador de magnitud es un circuito combinacional que compara dos números, a y b, y determina sus magnitudes relativas. La salida de la comparación se especifica por tres variables binarias que indican si $a > b$, $a = b$ y $a < b$.

El circuito combinacional por el método clásico para un comparador sería muy engorroso. Por ello se usará un procedimiento algorítmico que especifica un conjunto finito de pasos que, si se siguen, dan la solución a un problema. Considérense dos números a y b, con cuatro dígitos cada uno.

$$a = a_3\, a_2\, a_1\, a_0$$
$$b = b_3\, b_2\, b_1\, b_0,$$

en donde cada letra con subíndice representa uno de los dígitos del número. Los dos números son iguales si todos los pares de dígitos son iguales, o sea que:

$$a_3 = b_3$$
$$a_2 = b_2$$
$$a_1 = b_1$$
$$a_0 = b_0$$

Cuando los números son binarios, sólo tenemos dos dígitos que son 0 ó 1, y la relación de igualdad en forma lógica es:

$$X_i = a_i\, b_i + a_i'\, b_i' \quad i = 0, 1, 2, 3$$

$X_i = 1$ si y sólo si a_i y b_i son iguales. La salida cuando a y b son iguales en el circuito será $a = b$, y la salida será igual a 1 si todos los dígitos de a y b son iguales a 1. Si no es así, la salida es igual

a 0. Para que exista la condición de igualdad todas las variables X_i deben ser iguales a 1. Esto dicta una operación AND de todas las variables:

$$(a = b) = X_3 \bullet X_2 \bullet X_1 \bullet X_0$$

La variable binaria (a = b) es igual a 1 si y sólo si todos los pares de dígitos de los dos números son iguales.

Para determinar si a es mayor o menor que b, se inspeccionan las magnitudes relativas de pares de dígitos significativos, iniciando desde las posición más significativa. Si los dos dígitos son iguales, el par de dígitos de la siguiente posición significativa más baja se compara. Esta comparación continúa hasta que alcanza un par de dígitos desiguales. Si el dígito correspondiente de a es 1 y el de b es 0, se concluye que a > b. a < b será el caso contrario. Esto en álgebra de Boole se expresa así:

$$\begin{aligned}(a > b) &= a_3 \bullet b_3' + X_3 \bullet a_2 \bullet b_2' + X_3 \bullet X_2 \bullet a_1 \bullet b_1' \\ &\quad + X_3 \bullet X_2 \bullet X_1 \bullet a_0 \bullet b_0' \\ (a < b) &= a_3' \bullet b_3 + X_3 \bullet a_2' \bullet b_2 + X_3 \bullet X_2 \bullet a_1' \bullet b_1 \\ &\quad + X_3 \bullet X_2 \bullet X_1 \bullet a_0' \bullet b_0\end{aligned}$$

El diagrama lógico se muestra en la figura 7.7.1. El IC TTL 7485 es un comparador de magnitud de 4-bits. Este se muestra en la figura 7.7.2. Tiene tres entradas más para conectar comparadores en cascada. Las cuatro salidas X se generan con circuitos de equivalencia NOE y se aplican a una compuerta AND para dar la variable binaria de salida a = b. Las otras dos salidas utilizan las variables X para generar las funciones de Boole que se enlistan antes. Esta es una implementación de nivel múltiple y tiene un patrón regular. El procedimiento para obtener circuitos comparadores de magnitud para números binarios con más de cuatro bits debe ser obvio mediante este ejemplo.

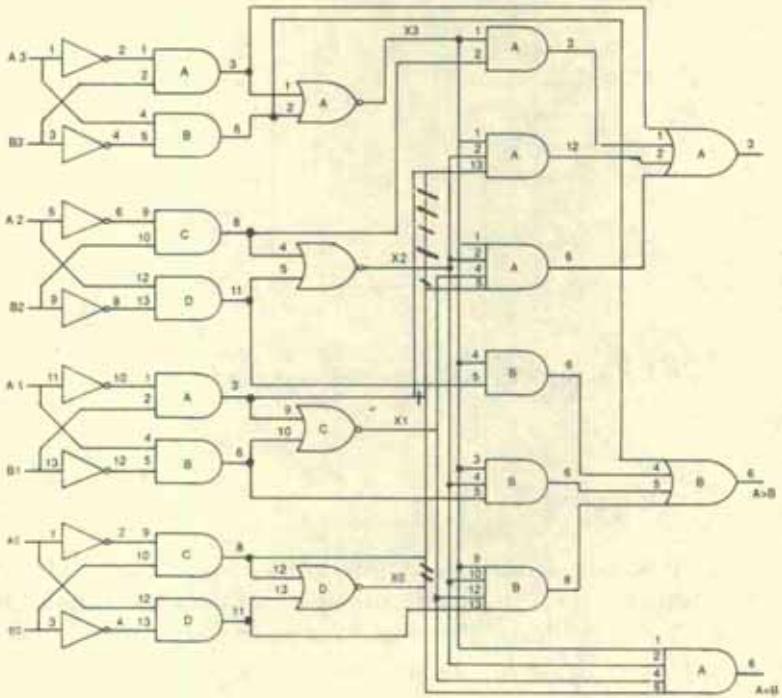


Figura 7.7.1. Diagrama lógico del comparador de magnitud

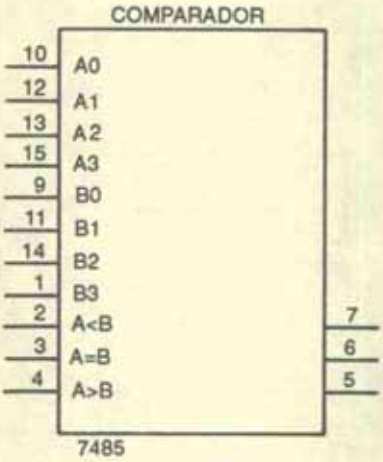


Figura 7.7.2. IC 7485. Comparador de magnitud.

7.8 Multiplexores

La multiplexión significa transmitir un gran número de unidades de información sobre un número más pequeño de canales o líneas. Un multiplexor digital es un circuito combinacional que selecciona información binaria de una de muchas líneas de entrada y la dirige a una sola línea de salida. La selección de una línea particular de entrada está controlada por un conjunto de líneas de selección. En forma normal, hay 2^n líneas de entrada y n líneas de selección cuyo valor binario determina qué entrada se selecciona. Un multiplexor de 4 líneas a una línea se muestra en la figura 7.8.1. Este multiplexor es similar al IC 74157. Cada una de las cuatro líneas de entrada, I0 a I3, se aplican a una entrada de una compuerta AND. Las líneas de selección S0 y S1

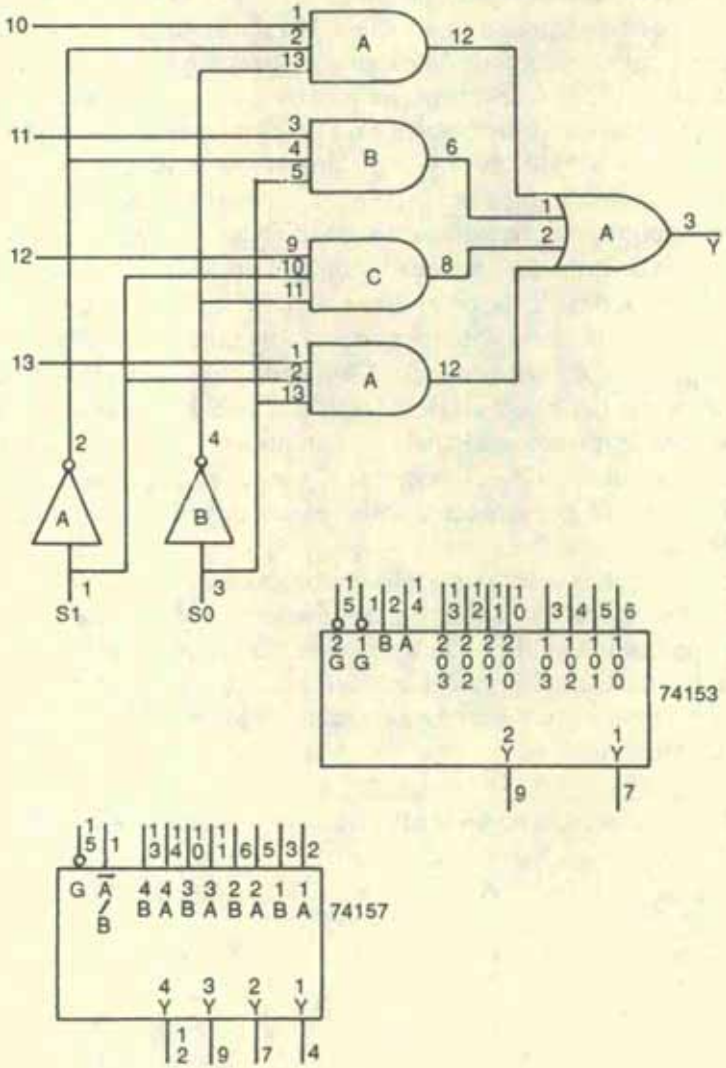


Figura 7.8.1. Multiplexor 4 líneas a una línea

se decodifican para seleccionar una compuerta AND particular. Cuando se utiliza un multiplexor en un sistema digital se representa por su diagrama de bloques. Un multiplexor también se conoce como selector de datos, ya que selecciona una de muchas entradas y dirige la información binaria a la línea de salida. Las compuertas AND e inversores en el multiplexor se asemejan a un circuito decodificador y, por supuesto, decodifican la selección de líneas de entrada. En general, un multiplexor de 2^n a una línea se construye mediante un decodificador n a 2^n, agregándole 2^n líneas de entrada, una a cada compuerta AND.

Las salidas de la compuerta AND se aplican a una sola compuerta OR para proporcionar una sola salida. El multiplexor se especifica por el número de 2^n líneas de entrada y salida. Esto implica que tiene n líneas de selección. También los multiplexores pueden tener una entrada de habilitación para controlar la operación de la unidad. La entrada E puede usarse para expandir dos o más multiplexores en un multiplexor digital con un número más grande de salidas.

En algunos casos dos o más multiplexores se encapsulan en un solo paquete IC. Las entradas de selección y de habilitación en una unidad múltiple de IC pueden ser comunes a todos los multiplexores. La figura 7.8.2 muestra un multiplexor cuádruple de dos líneas a una línea muy similar al 74157. Su tabla de verdad es la siguiente:

Tabla de verdad de un multiplexor de 2 líneas a una línea

ENTRADAS		SALIDA
E	S	Y
1	X	0
0	1	Selecciona a
0	0	Selecciona b

La figura 7.8.2 muestra cuatro multiplexores en un solo circuito, cada uno capaz de seleccionar una de las dos líneas de entrada.

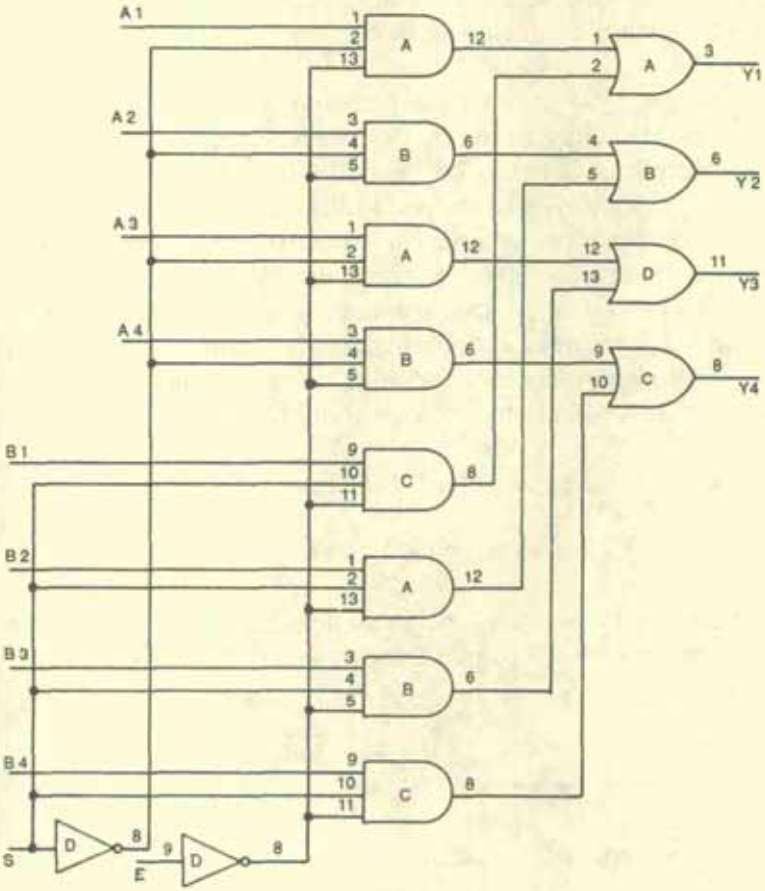


Figura 7.8.2. Multiplexor dos líneas a una línea

La salida Y puede ser igual a *a* o a *b*, dependiendo de S, y así sucesivamente. Los multiplexores estarán habilitados cuando E es igual a 0 y se deshabilitarán cuando E sea igual a 1.

El multiplexor tiene muchas aplicaciones. Se puede usar para conectar dos o más fuentes a un solo destino. Es útil para construir un sistema de bus común. Uno de sus usos es implementar cualquier función de Boole.

Aplicaciones. Se mostró que con un decodificador se puede implementar una función de Boole empleando un compuerta OR externa. En la figura 7.8.1 el multiplexor funciona como un decodificador con una compuerta OR. Los mintérminos de salida del decodificador seleccionados se controlarán con las líneas de entrada. Si se tiene una función de Boole de n+1 variables, se toman n de esas variables y se conectan a las líneas de selección de un multiplexor. La variable que queda se pone en las entradas del multiplexor. En esta forma es posible generar cualquier función de n+1 variables con un multiplexor de 2^n a 1.

Ejemplo 7.8.1. Implementar la función:

$$F(a, b, c) = \sum m(1, 3, 5, 6)$$

con un multiplexor. La función se puede implementar con un multiplexor de 4-a-1 como se muestra en la figura 7.8.3.

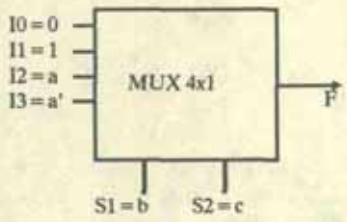


Tabla de verdad de la función				
mintérmino	entradas			salidas
	a	b	c	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Dos de las variables se aplican a las líneas de selección. Para implementar cualquier función de Boole de n variables con un multiplexor de 2^{n-1} a 1 primero se expresará la función en su forma de suma de mintérminos. La secuencia ordenada de las variables será a b c d . . . , donde b c d . . . son las n-1 variables que se conectarán a las líneas de selección del multiplexor, con B al bit de selección más significativo.

Se conectará a' a la primera mitad de mintérminos y a sin complementar a la segunda mitad. Por ejemplo, en una función de tres variables a, b, y c se tienen ocho mintérminos. La variable a se complementa desde el mintérmino 0 hasta el mintérmino tres y no se complementa en los restantes. Se enlistan las entradas del multiplexor, y bajo ellas los mintérminos en dos renglones.

	I0	I1	I2	I3
a'	0	1	2	3
a	4	5	6	7

En el primer renglón están los mintérminos donde a está complementada. En el segundo renglón están los mintérminos, donde a no está complementada. Se les pone un asterisco a los mintérminos de la función y se inspecciona por separado cada columna:

	I0	I1	I2	I3
a'	0	1*	2	3*
a	4	5*	6*	7

Si los dos mintérminos en una columna no están señalados se aplicará un cero a la entrada correspondiente del multiplexor. Si los dos mintérminos están señalados se aplica un 1 a la entrada correspondiente del multiplexor. Si el mjntérmino inferior está señalado y el superior no lo está se aplica a a la entrada correspondiente del multiplexor. Si el mintérmino superior está seña-

lado y el inferior no se aplica a' a la entrada del multiplexor correspondiente. El orden de a, b y c no importa si se tiene cuidado al hacer la tabla de implementación. Como ejemplo de ello se hará el problema anterior de nuevo en orden diferente:

$$F(a, b, c) = \sum m(1, 3, 5, 6)$$

	I0	I1	I2	I3
c'	0	2	4	6*
c	1*	3*	5*	7

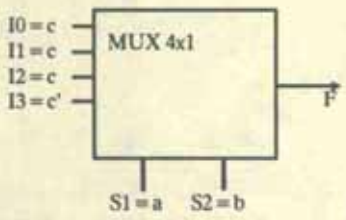


Tabla de verdad de la función

mintérmino	entradas a	b	c	salidas F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

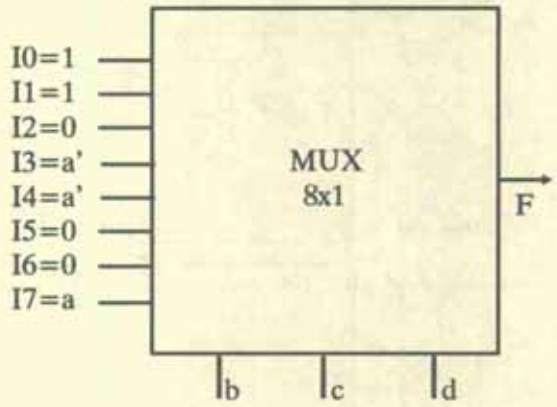
Es posible usar cualquier variable única de la función para usarse en la entrada del multiplexor. Todas las demás se aplican a las líneas de selección.

Ejemplo 7.8.2. Implemente la siguiente función con un multiplexor.

$$F(a, b, c, d) = \sum m(0, 1, 3, 4, 8, 9, 15)$$

Esta es una función de cuatro variables. Por tanto, se necesita un multiplexor con tres líneas de selección de ocho entradas. Se aplicará b, c y d a las líneas de selección. La implementación se muestra a continuación.

	I0	I1	I2	I3	I4	I5	I6	I7
a'	0*	1*	2	3*	4*	5	6	7
a	8*	9*	10	11	12	13	14	15*



Los multiplexores y los decodificadores son útiles en circuitos pequeños, pero para circuitos combinacionales más grandes se utilizarán las memorias y los PLAs.

7.9 Demultiplexores

Algunos ICs decodificadores se construyen con compuertas NAND. Algunos incluyen una o más entradas de habilitación para controlar la operación del circuito. La figura 7.9.1 muestra un decodificador de 2-a-4 líneas con una entrada de control o de habilitación. Está construido con compuertas NAND. La tabla de verdad para este circuito se muestra en la figura 7.9.2.

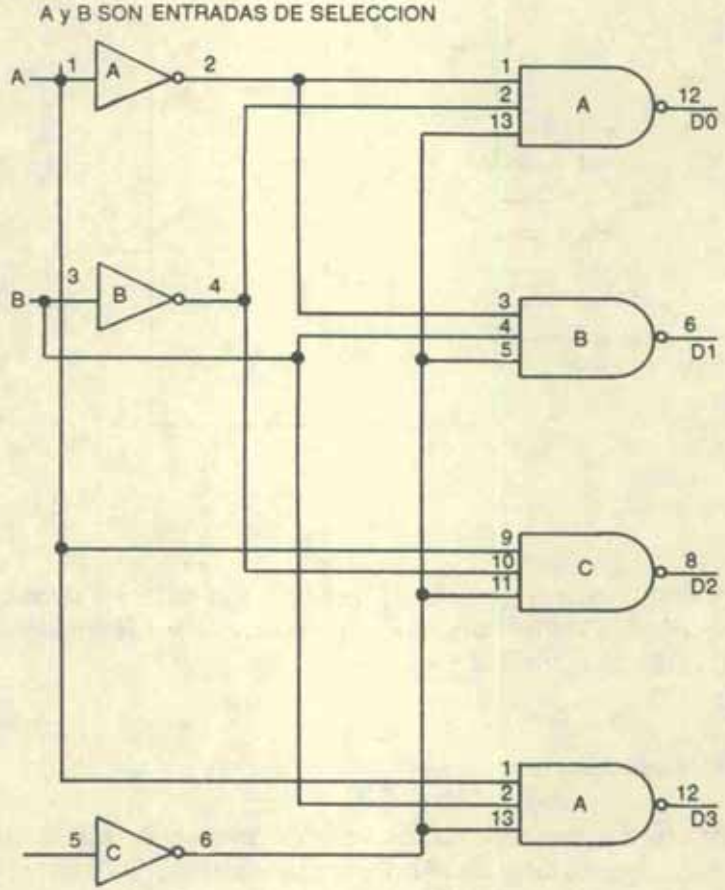


Figura 7.9.1. Diagrama lógico del demultiplexor

Todas las salidas son iguales a 1 si la entrada de habilitación E es igual a 1, sin importar el valor de a y b. Cuando E es 0, el circuito opera como un decodificador con salidas complementadas. La tabla de verdad enlista esas condiciones.

El diagrama de bloques del decodificador se muestra en la figura 7.9.2 (a). El círculo pequeño en la entrada E indica que el decodificador está habilitado cuando E es igual a 0. Los círculos a las salidas indican que todas las líneas están complementadas.

Figura 7.9.2 (a) Decodificador con habilitación

Un decodificador con una entrada de habilitación E puede funcionar como un demultiplexor. Un demultiplexor es un circuito que recibe información en una sola línea y transmite esta información en una de 2^n líneas de salida posibles. La selección de una línea específica de salida está controlada por el valor binario de las n líneas de selección. Ver la figura 7.9.2 (b). El decodificador puede funcionar como un demultiplexor si la línea E se toma como entrada de datos y las líneas a y b como las de selección. La variable de entrada E tiene una trayectoria a todas las cuatro salidas, pero la información de entrada se dirige a una sola de las líneas de salida dependiendo del valor binario de a y b. Esto se verifica en la tabla de verdad del decodificador. Por ejemplo, si ab = 10, la salida D2 será la misma que el valor de entrada E, en tanto que las otras salidas se mantienen en 1. Por tanto, un decodificador con entrada E es un decodificador/demultiplexor al mismo tiempo.

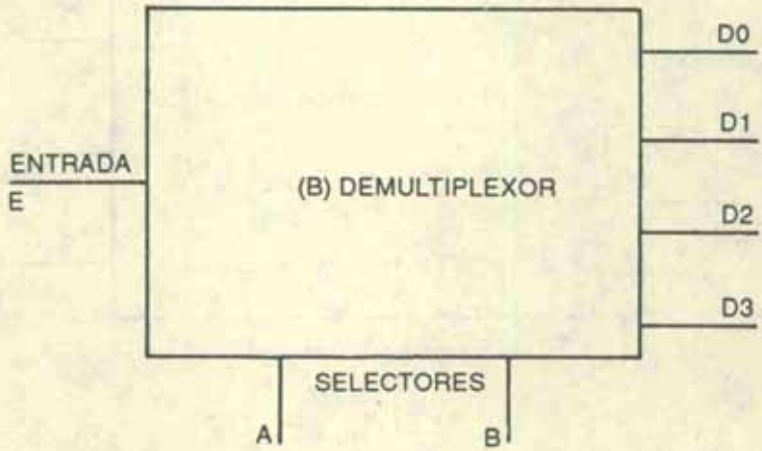


Figura 7.9.2 (b) Demultiplexor

Los circuitos decodificadores/demultiplexores pueden conectarse juntos para formar un decodificador más grande. La figura 7.9.3 muestra dos decodificadores de 3 x 8 con entradas de

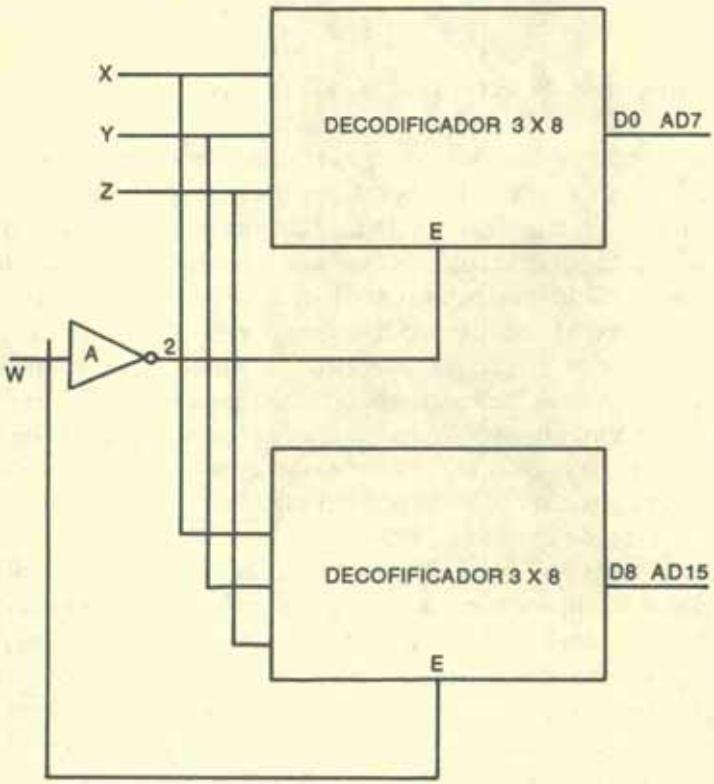


Figura 7.9.3 Decodificador de 4 x 16 construido con dos decodificacores de 3 x 8

habilitación conectadas para formar un decodificador de 4 x 16. Si w = 0. el decodificador 1 es el que está activado. Las salidas del decodificador 2 son todas cero y las ocho salidas superiores generan mintérminos de 0 0 0 0 a 0 1 1 1. Si w = 1, las condiciones se invierten. El decodificador inferior genera los mintérminos desde 1 0 0 0 hasta 1 1 1 1 y las salidas del decodificador 1 son todas 0.

7.10 Memorias de lectura solamente (ROMs)

Ya se vio que un decodificador genera los 2^n mintérminos de las n variables de entrada. Por la inserción de compuertas OR para la suma de los mintérminos de las funciones de Boole se tuvo capacidad de generar cualquier circuito combinacional deseado. Una memoria de sólo lectura (ROM) es un dispositivo que incluye tanto el decodificador como las compuertas OR dentro de un solo paquete IC. Las conexiones entre las salidas del decodificador y las entradas de las compuertas OR pueden especificarse para cada configuración particular por la "programación" de la ROM. La ROM se usa para implementar un circuito combinacional complejo en un paquete IC y, en ese caso, se eliminan todos los alambres de interconexión.

Una ROM en forma esencial es un dispositivo de memoria en el que se almacena un conjunto fijo de información binaria. La información binaria primero debe especificarla el usuario y entonces se inserta en la unidad para formar el patrón requerido de interconexión. Las ROM se obtienen con eslabones internos especiales que pueden fusionarse o romperse. La interconexión deseada para una aplicación particular requiere que se fusionen ciertos eslabones para formar las trayectorias del circuito necesario. Una vez que se establece un patrón para una ROM, permanece fijo aun cuando se deje de suministrar energía al

circuito. La figura 7.10.1 muestra el diagrama de bloques de una memoria ROM.

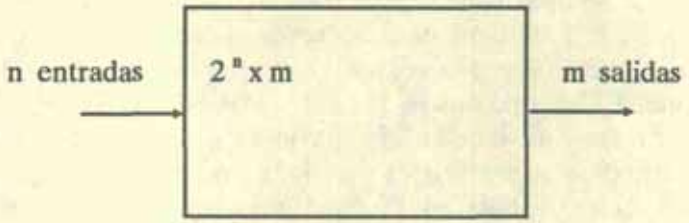


Figura 7.10.1. Diagrama de bloques de una ROM

Consta de n líneas de entrada y m líneas de salida. Cada combinación de bits de las variables de entrada se llama dirección. Cada combinación de bits que sale de las líneas de salida se conoce como palabra. El número de bits por palabra es igual al número de líneas de salida m. Una dirección es en esencia un número binario que denota uno de los mintérminos de las n variables. El número de direcciones distintas posibles con n variables de entrada es 2^n. Una palabra de salida puede seleccionarse por una dirección única, si hay 2^n direcciones distintas en una ROM. Entonces hay 2^n palabras distintas que se almacenan en la unidad. La palabra a la salida dependerá del valor de dirección aplicado a las entradas. Una ROM se caracteriza por el número de palabras 2^n y el número de bits por palabra m.

Considérese una ROM de 32 x 8. La unidad consta de 32 palabras de 8 bits cada una, o sea que hay 8 líneas de salida y 32 palabras almacenadas en la unidad. La palabra seleccionada es determinada por las cinco líneas de entrada. Si la dirección de

entrada es 00000 se selecciona la palabra que está en la dirección 0 y ésta aparece en la salida.

Hay otras 30 direcciones que pueden seleccionar las otras 30 palabras. Algunas veces las ROMs se especifican por el número total de bits que contienen, el cual es 2^n x m. Por ejemplo, una ROM de 2048 bits puede organizarse en 512 palabras de cuatro bits cada una. Esto significa que la unidad tiene cuatro líneas de salida y nueve líneas de entrada para especificar $2^9 = 512$ palabras. El total de bits es 512 x 4 = 2048. En su interior, la ROM es un circuito combinacional con compuertas AND conectadas como un decodificador y un número de compuertas OR igual al número de salidas de la memoria. La figura 7.10.2 muestra la construcción lógica interna de una ROM de 32 x 4.

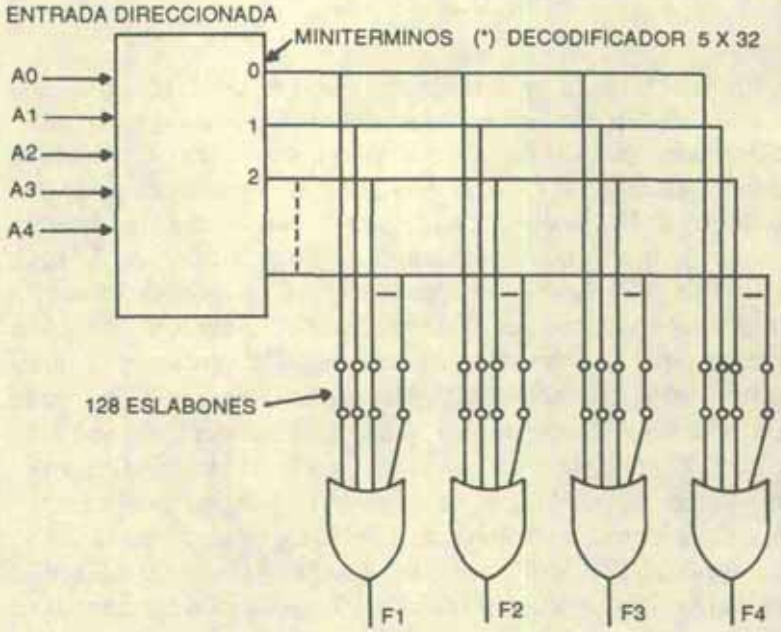


Figura 7.10.2. Diagrama lógico de una ROM de 32 x 4

Las cinco variables de entrada están decodificadas en 32 líneas mediante 32 compuertas AND y cinco inversores. Cada salida del decodificador representa uno de los mintérminos en una función de cinco variables. Cada una de las 32 direcciones selecciona una y sólo una salida del decodificador. La dirección es un número de cinco bits aplicado a las entradas, y el mintérmino seleccionado de salida del decodificador es el marcado con el número decimal equivalente. Las 32 salidas del decodificador están conectadas a través de 32 eslabones a cada compuerta OR. No todos los eslabones se muestran en el diagrama, pero en la realidad cada compuerta OR tiene 32 entradas, y cada entrada va a través de un eslabón que puede romperse según se desee.

La ROM es una implementación de dos niveles en la forma de suma de mintérminos. El segundo nivel por lo común es una conexión de lógica alambrada para facilitar la fusión de eslabones. Mediante el diagrama lógico de la ROM es claro que cada salida proporciona la suma de todos los mintérminos de las n variables de entrada. Recuérdese que cualquier función de Boole puede expresarse en la forma de suma de mintérminos.

Por la ruptura de los eslabones de los mintérminos que no se incluyen en la función, cada salida ROM puede hacerse que represente la función de Boole de una de las variables de entrada en el circuito combinacional. Para un circuito combinacional de n entradas y m salidas se necesita una ROM de 2^n x m. La apertura de los eslabones se conoce como programación de la ROM. El diseñador necesita especificar sólo una tabla de programa de la ROM que da la información para las trayectorias requeridas en la ROM. La programación es un procedimiento de hardware que sigue las especificaciones que se enlistan en la tabla del programa.

Ejemplo 7.10.1. La siguiente tabla de verdad especifica un circuito combinacional con dos entradas y dos salidas:

a1	a0	F1	F2
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	0

Las funciones de Boole pueden expresarse en forma de mintérminos:

$$F1\ (a1, a0) = \sum m\ (1, 2, 3)$$
$$F2\ (a1, a0) = \sum m\ (0, 2)$$

Cuando se implementa un circuito combinacional mediante una ROM, las funciones deben expresarse en suma de mintérminos o, mucho mejor, en una tabla de verdad. Si las funciones de salida se simplifican, se encontrará que el circuito sólo necesita una compuerta OR y un inversor. En forma obvia, este es un circuito combinacional demasiado simple para implementarse con una ROM. La ventaja de una ROM se hace patente en los circuitos combinacionales complejos. Este ejemplo simple muestra el procedimiento y no debe considerarse en una situación práctica. La ROM que implementa al circuito combinacional debe tener dos entradas y dos salidas, de modo que su tamaño debe ser de 4 x 2. La figura 7.10.3 muestra la construcción interna de dicha ROM y qué eslabones se rompieron de acuerdo con F1 y F2.

En la práctica, cuando se diseña un circuito mediante una ROM no es necesario mostrar las conexiones internas de la compuerta y los eslabones. Sólo hay que especificar la ROM particular o su número de designación, y su tabla de verdad da la información para su programación.

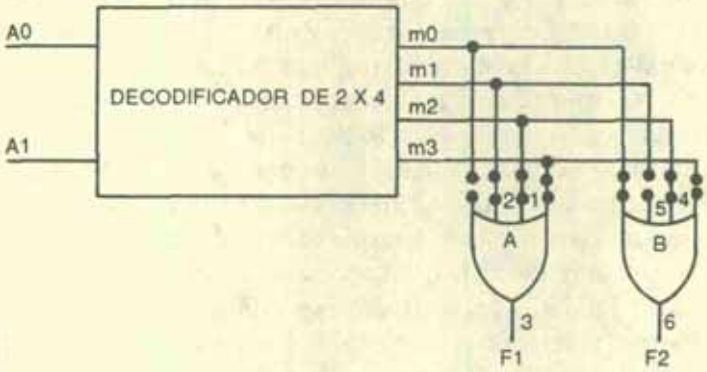


Figura 7.10.3. En la ROM se indican los eslabones que se rompieron de acuerdo con el valor que se obtuvo para F1 y F2

Ejemplo 7.10.2. Diseñe un circuito combinacional usando una ROM. El circuito acepta un número binario de 3 bits y genera un número binario de salida igual al cuadrado del número de entrada. Primero se obtiene la tabla de verdad del circuito combinacional:

Tabla de verdad

Entradas			Salidas						Decimal
a2	a1	a0	b5	b4	b3	b2	b1	b0	
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

En la mayoría de los casos esto es todo lo que se necesita. En algunos casos puede ajustarse una tabla de verdad más pequeña para la ROM. Se requieren tres entradas y seis salidas para acomodar todos los números posibles. Se puede observar que la salida b0 siempre es igual a la entrada a0, de modo que no se necesita generar b0 con una ROM ya que es igual a una variable de entrada. Además, la salida b1 siempre es 0, de modo que su salida siempre se conoce. En realidad sólo es necesario generar cuatro salidas con la ROM. Las otras dos se obtienen con facilidad. El tamaño mínimo de la ROM necesaria debe tener tres entradas y cuatro salidas. Tres entradas especifican ocho palabras, de modo que el tamaño de la ROM debe ser 8 x 4. La figura 7.10.4 muestra la implementación de este circuito.

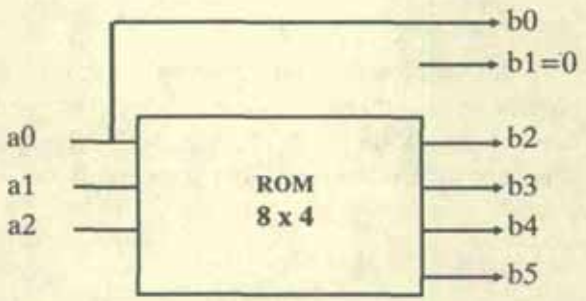


Figura 7.10.4. Circuito que genera el cuadrado de la entrada implementado con una ROM

Las trayectorias requeridas en una ROM pueden programarse en dos formas diferentes. La primera la hace el fabricante durante el último proceso de producción de la unidad. Se requiere que el cliente llene la tabla de verdad que se presentará en la forma que lo indique el fabricante. El fabricante hace la máscara correspondiente de las trayectorias para producir los

unos y ceros requeridos por el cliente. Este tipo de programación es económica sólo si van a fabricarse grandes cantidades de la misma configuración ROM.

Para pequeñas cantidades se utiliza la PROM. La PROM se programa rompiendo los eslabones por la aplicación de pulsos de corriente a través de las terminales de entrada. Un eslabón roto define un estado binario y un eslabón sin romper el otro. Hay unidades denominadas programadores PROM para facilitar el procedimiento, aunque todos estos procedimientos son de hardware. El procedimiento de hardware para programar las ROM o PROM es irreversible, y una vez programados los patrones fijos son permanentes y no pueden alterarse.

Las EPROM pueden reestructurarse a su valor inicial, todos 0 o todos 1, aun cuando se hayan cambiado previamente. Cuando una EPROM se coloca bajo una luz ultravioleta especial por un periodo de tiempo, la radiación de onda corta descarga las compuertas internas que sirven como contactos. Así, la EPROM regresa a su estado inicial y puede reprogramarse. Algunas se borran con señales eléctricas en lugar de luz ultravioleta y se llaman EEPROM.

Las ROM se utilizan para implementar circuitos combinacionales complejos en forma directa desde sus tablas de verdad. Son útiles para convertir de un código binario a otro, para funciones aritméticas y para otras muchas aplicaciones.

7.11 Arreglos lógicos programables (PLAS)

Cuando se va a diseñar un circuito combinacional en el que las condiciones *no importa* son muchas, utilizar una ROM es un desperdicio. Para estos casos se utilizan los circuitos LSI llamados PLAS. Un PLA es similar a una ROM. .El FPLA es la versión programable en el campo.

Como su nombre lo dice, el PLA es un dispositivo de arreglos lógicos diseñado para implementar expresiones lógicas aleatorias en la forma de suma de productos de manera similar a la ROM. Sin embargo, el PLA difiere de la ROM considerablemente en la estructura. Esto se puede apreciar en la figura 7.11.1:

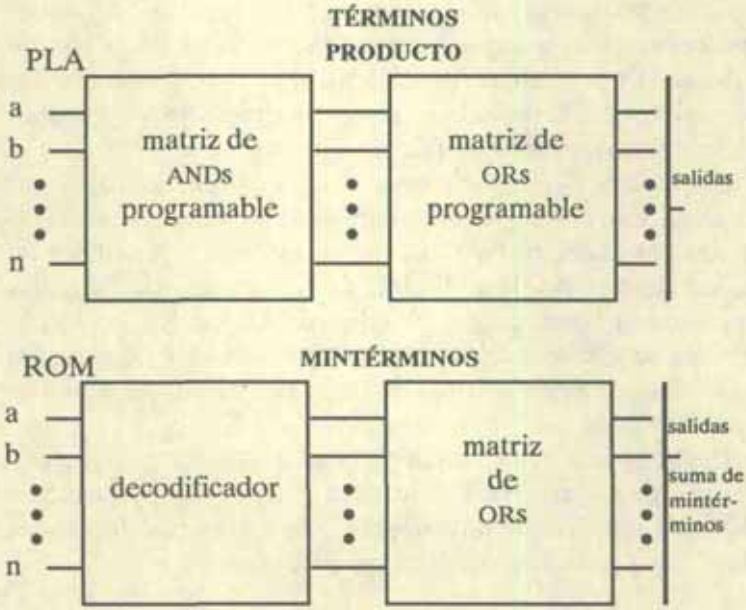


Figura 7.11.1. Comparación de estructuras entre un PLA y una ROM

La expresión lógica para la ROM sería:

$$F = a'b'c' + a'b'c + a'bc + ab'c' + abc$$

La implementación de F se muestra en las figuras 7.11.2 (a) y 7.11.2 (b) utilizando una ROM y un PLA.

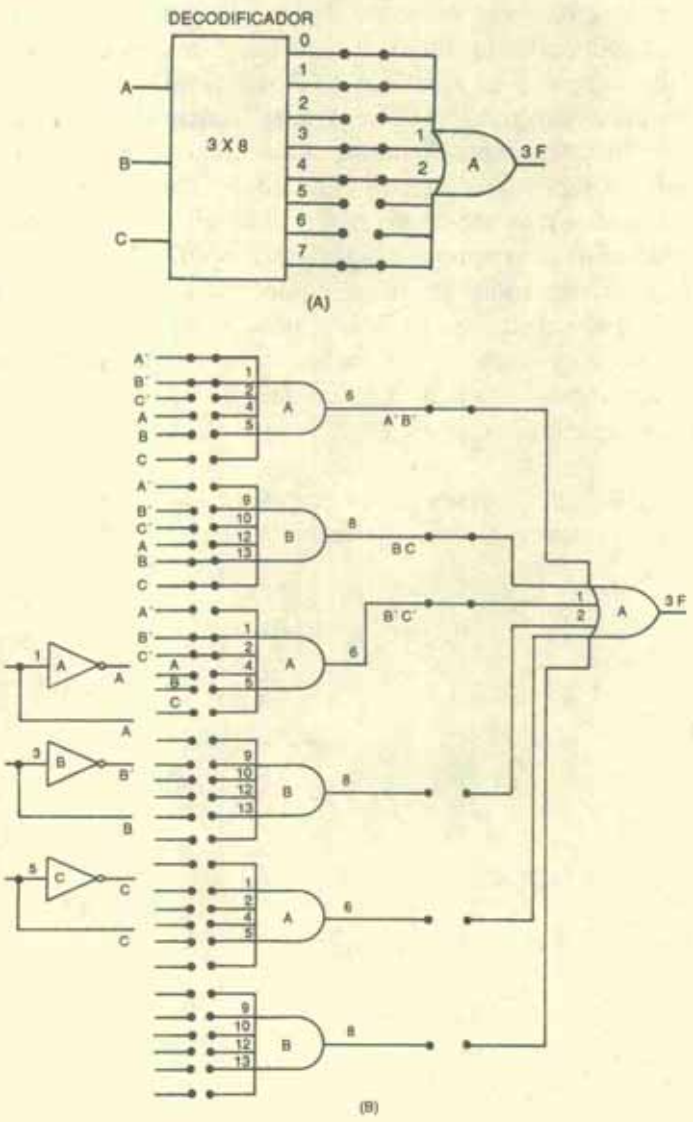


Figura 7.11.2. (a) con una ROM. (b) con un PLA

Una gran diferencia entre estas dos estructuras está en los decodificadores de entrada. La ROM tiene un decodificador de entrada completo. El PLA tiene un decodificador en matriz con compuertas AND programables. Esa diferencia elimina el almacenamiento ineficiente de mintérminos no necesarios que en el caso de la ROM no se pueden eliminar debido a su exhaustivo decodificador de entrada de n a 2^n. Por ello un PLA permite programar mejor expresiones lógicas simplificadas, en vez de usar las formas canónicas que se requieren cuando se utiliza una ROM. Debido a esto, un PLA típico puede acomodar muchas más entradas que una ROM. Para ilustrar la implementación de una expresión lógica aleatoria utilizando un PLA y una ROM, véase el siguiente ejemplo.

Ejemplo 7.11.1. Se desea poner en práctica la expresión lógica que se muestra en el siguiente mapa:

a \ b c	00	01	11	10
0	1	1	1	
1	1		1	

$$F(a, b, c) = \sum m(0, 1, 3, 4, 7)$$

La expresión lógica para el PLA sería:

$$F = a'b' + bc + b'c'$$

La expresión lógica para la ROM sería:

$$F = a'b'c' + a'b'c + a'bc + ab'c' + abc$$

La implementación de F se muestra en la figura 7.11.2 (a) y la figura 7.11.2 (b) utilizando una ROM y un PLA.

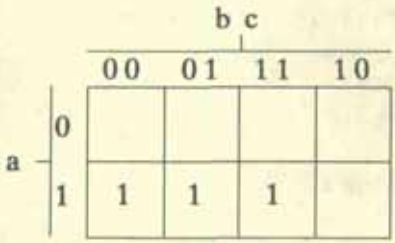
Los PLAs se miden o se especifican de la siguiente manera: a x b x c; donde a es el número de entradas, b es el número total de términos producto —número de entradas que cada compuerta OR tiene— y c es el número de salidas —las diferentes expresiones lógicas obtenidas.

El 82S100 es un FPLA. Sus especificaciones son: 16 x 48 x 8. Esto implica que este FPLA es capaz de generar ocho diferentes salidas lógicas, cada una con un máximo de 48 términos producto únicos, hechos con cualquier combinación de 16 variables de entrada.

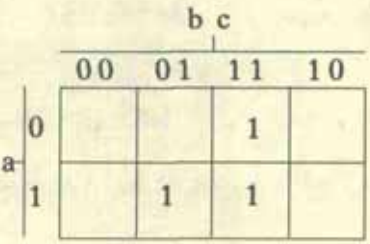
Ejemplo 7.11.2. Diseñar con un PLA un circuito definido por:

$$F1 = \sum m\,(4, 5, 7)$$
$$F2 = \sum m\,(3, 5, 7)$$

Se reduce con el mapa de Karnaugh:



$F1 = a\,b' + a\,c$



$F2 = a\,c + b\,c$

Posteriormente se hace una tabla para programar el PLA:

	Término producto	Entradas a b c	Salidas F1 F2	
a b'	1	1 0 -	1 -	
a c	2	1 - 1	1 1	
b c	3	- 1 1	- 1	
			T T	T/C

Así es como se mandan las especificaciones al fabricante para que él la programe. T significa que se utilizará la función tal y como está. C significa que se utilizará la función complementada. En la tabla del PLA se pone un guion a la entrada que se elimina. En las salidas, se marcan con un guión los términos productos que no se utilizan.

Ejemplo. Diseñar con un PLA un circuito que se define por:

$$F1 = \sum m\,(3, 5, 6, 7)$$
$$F2 = \sum m\,(0, 2, 4, 7)$$

Con los mapas de Karnaugh se obtiene:

$$F1 = (b'\,c' + a'\,c' + a'\,b')'$$
$$F2 = b'\,c' + a'.c' + a\,b\,c$$

Se utiliza F1' con el objeto de tener más compuertas en común. Se obtienen 4 términos producto distintos. La tabla es la siguiente:

	Término producto	Entradas a b c	Salidas F1 F2		
b' c'	1	- 0 0	1	1	
a' c'	2	0 - 0	1	1	
a' b'	3	0 0 -	1	-	
abc	4	1 1 1	-	1	
			C	T	T/C

7.12 Conclusiones

Con este capítulo finaliza todo lo que se refiere al estudio de los circuitos combinacionales. Lo primero que se hizo fue presentar las compuertas básicas, después cómo diseñar con ellas circuitos especializados y, finalmente, cómo diseñar sistemas con circuitos combinacionales de mediana escala. Se mostró también el funcionamiento de una memoria ROM que es un circuito combinacional de propósito general. Puede decirse que es un circuito combinacional universal pues con él se puede diseñar cualquier circuito combinacional.

Al finalizar este capítulo se recomienda realizar la práctica número diez del Apéndice B, dedicada a los circuitos combinacionales.

7.13 Ejercicios

Diseñar un circuito que realice la operación lógica OR, AND y OE de dos números a y b de 2 bits cada uno, los números:

$$a = a1\ a2 \text{ y } b = b1\ b2.$$

Por ejemplo, si a = 01 y b = 00, entonces la salida será Z = 01 cuando se realice la operación OR.

Diseñar este circuito con:

- Decodificadores,
- Multiplexores,
- ROMs y
- PLAs.

Apéndice A

El objetivo de este libro, texto para un curso, es que el lector adquiera los conocimientos tanto teóricos como prácticos en lo que se refiere a los circuitos digitales. Por ello se requiere que el lector realice varias prácticas con circuitos integrados digitales.

Las dos familias de circuitos digitales más importantes son la TTL (*Transistor-Transistor Logic*) y la CMOS (*Complementary Metal Oxide Semiconductor*). Estos circuitos necesitan que se les suministre corriente directa. Esta se puede obtener de una batería. Sin embargo, se desgasta rápidamente. A la larga resulta mejor y más económico utilizar una fuente regulada. Por eso se ha incluido este apéndice donde se muestra cómo diseñar una fuente regulada de bajo costo y alto rendimiento. Se incluye en el diagrama final el diseño de un oscilador para ser utilizado cuando se realicen experimentos con circuitos secuenciales.

En esta obra se trabajará al inicio con la familia TTL por ser más fácil de manejar para los lectores principiantes.

Fuentes de alimentación

El voltaje que suministra la Comisión Federal de Electricidad (CFE) es de 120 voltios de corriente alterna. A lo largo de este texto se muestran técnicas de diseño con circuitos digitales. Los circuitos digitales trabajan con voltaje de corriente continua. Los circuitos integrados TTL que se utilizan en las prácticas trabajan a 5 voltios de corriente continua; por tanto, se requerirá convertir el voltaje de corriente alterna a voltaje de corriente continua. Esto se logra diseñando una fuente de alimentación.

El circuito consta de un transformador, un puente rectificador, un capacitor de filtro, un circuito regulador de voltaje y un capacitor de salida. Será conveniente, además, poner a la entrada un fusible y un interruptor. El circuito se muestra en la figura A-1.

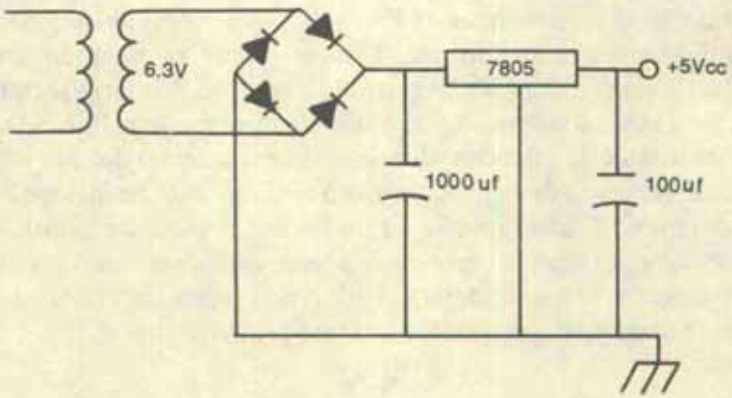


Figura A-1. Fuente de alimentación

La entrada del transformador recibe una corriente alterna con un voltaje de 120 voltios, aproximadamente. El transformador sirve para reducir el voltaje de 120 a 6 voltios de corriente alterna. En seguida se utiliza un circuito rectificador de onda completa, que convierte la corriente alterna en corriente continua pulsante. La forma de onda a la salida del rectificador de onda completa se muestra en la figura A-2.

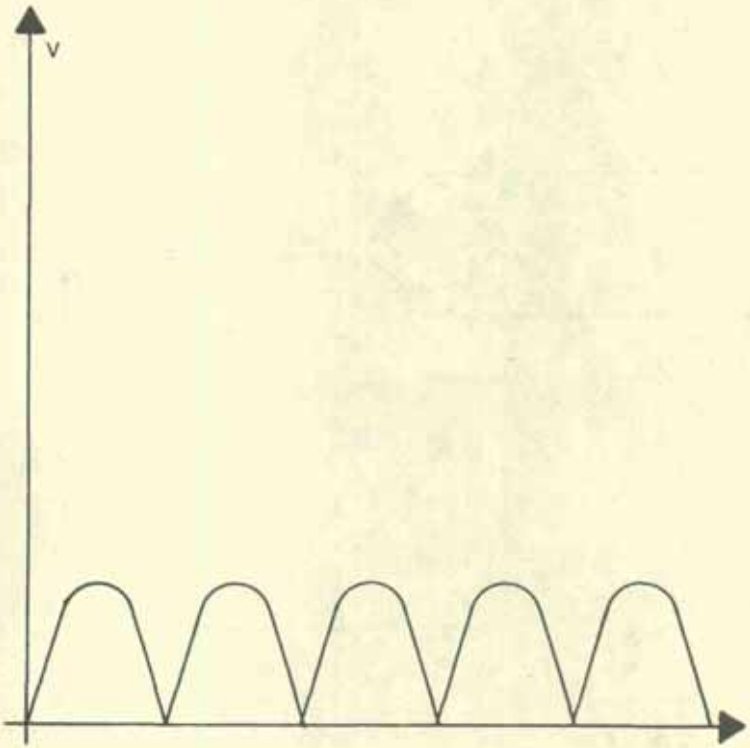


Figura A-2. Voltaje pulsante a la salida del rectificador

Los diodos en este caso son el 1N4003, aunque también existen los cuatro diodos integrados en un solo paquete. Más adelante se explicará en detalle cómo funciona cada uno de estos dispositivos.

El circuito funciona de la siguiente manera: durante los ciclos positivos de voltaje de corriente alterna (Ver figura A-3) la corriente fluirá por D1, RL y D3, porque el diodo es un dispositivo que sólo conduce cuando está polarizado directamente (Ver figura A-4). RL (Resistencia de carga).

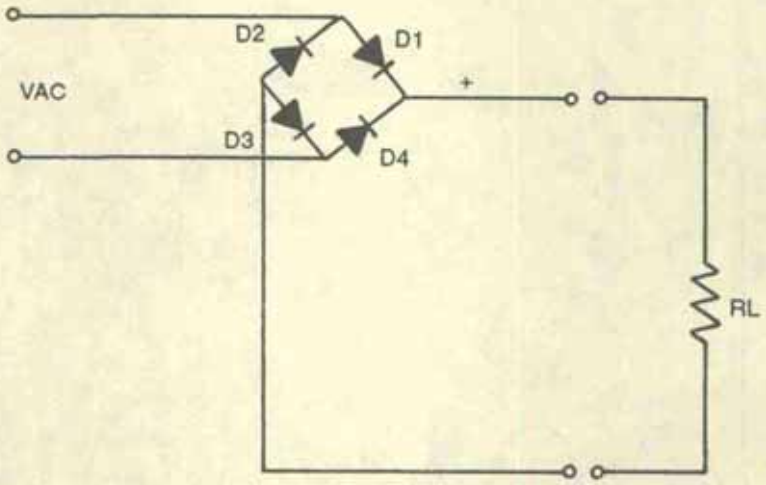


Figura A-3. Circuito rectificador con RL

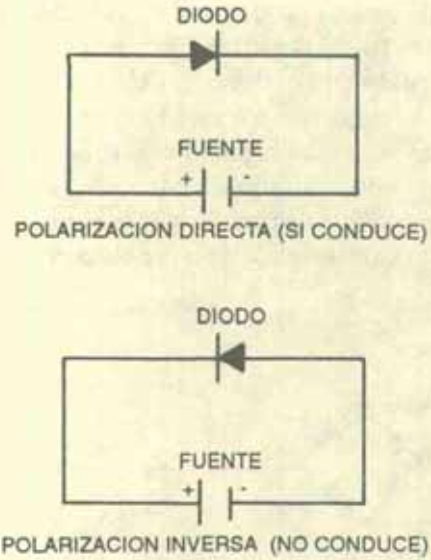


Figura A-4. Polarización del diodo

Por lo tanto por los diodos D2 y D4 no hay flujo de corriente porque están polarizados inversamente. Cuando el voltaje de corriente alterna es negativo, la corriente fluirá por D4, R y D2 mientras que los diodos D1 y D3 no conducen. La corriente pasará por RL siempre en el mismo sentido y, por lo tanto, hay rectificación de onda completa.

Después del circuito rectificador la fuente necesitará un filtro, el cual consistirá en un capacitor cuya función será disminuir la ondulación y aumentar la tensión de salida.

La ondulación o voltaje de rizo es un inevitable pequeño voltaje variable que es producto de un circuito rectificador, el cual disminuye a través del filtro. El capacitor almacena energía durante el periodo de conducción del rectificador para entregarla a la carga cuando no hay conducción.

De esta manera si el capacitor es suficientemente grande nunca se interrumpe el flujo de corriente. A este voltaje que varía se le llama voltaje de rizo. Una representación aproximada de la onda de tensión de salida de la fuente con el filtro se muestra en la figura A-5.

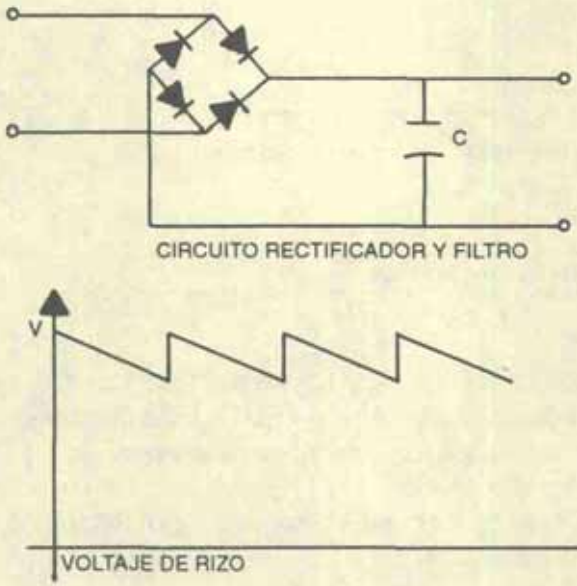


Figura A-5. Circuito rectificador y filtro

Para terminar con el diseño de la fuente sólo falta agregar un regulador de voltaje a cinco voltios y un filtro más que consiste en un capacitor de tantalio de 10 microfaradios para aumentar la estabilidad.

Los circuitos TTL son muy sensibles a las variaciones de voltaje. Esto no representa problemas si se utiliza el regulador de voltajes, epecíficamente el regulador de voltaje 7805 o LM340. Con este regulador se obtienen voltajes muy precisos. Para esta fuente se recomienda el encapsulado de metal. Estos reguladores tienen protección interna de sobrecarga y, como se habrá notado, no se requiere de muchos componentes externos. Es muy recomendable para suministro de voltaje en circuitos TTL. Este regulador de voltaje deberá tener a la entrada un voltaje mínimo de 7.5 voltios y uno máximo de 12 voltios.

Sería útil integrar a la fuente un oscilador. Si se desea se puede armar el circuito que se muestra en la figura A-6, pues incluye circuitos integrados cuyo funcionamiento se explicará en el tomo dos de esta obra. La frecuencia que se obtiene es de un ciclo por segundo.

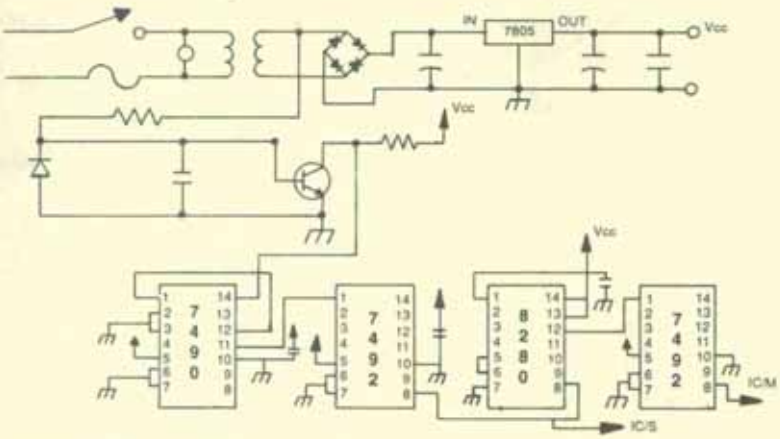


Figura A-6. Fuente y oscilador

Apéndice B

Prácticas de circuitos combinacionales

PRACTICA UNO. Compuertas básicas

Investigar en los manuales de TTL cuáles son los circuitos que realizan las siguientes operaciones lógicas: NAND, NOR, NOT, AND, OR, XOR (con dos entradas solamente).

Conectar cada uno de los circuitos a tierra y Vcc como se indica en el manual y probar una de las compuertas de cada circuito integrado. A la salida del circuito deberá conectarse un LED con una resistencia en serie cuyo valor será de 180 ohms a 1K ohm a 1/4 de watt.

Considérense los dos valores de una señal binaria, 0 y 1. En circuitos 1 será igual a H igual a 5 voltios. Cero, 0 será igual a L o a tierra.

Obtener las tablas de verdad para cada una de las funciones lógicas y armar los circuito que se muestran en la figura 2.2.7.

PRACTICA DOS. Problema de aplicación

Diseñar un circuito con tres entradas y una salida. Las entradas son un número binario de tres bits del 0 al 7. A la salida del circuito se conectará un LED que se prenderá cuando a la entrada del circuito este un número múltiplo de dos o incluso el dos. Utilizar sólo compuertas NAND 74LS00 de dos entradas.

PRACTICA TRES. Compuertas de colector abierto

Implementar las siguiente función con compuertas de colector abierto. $Y = (a\ b)'\ (c\ d)' = (a\ b + c\ d)'$. Recordar que las tres aplicaciones principales de las compuertas de colector abierto son:

(a) Aumentar el *Fan-Out* de un circuito.
(b) Realizar lógica alambrada.
(c) Construir un sistema de bus común.

PRACTICA CUATRO. Lógica de tres estados

La característica importante de la lógica de tres estados es que sus salidas pueden conectarse entre si, sin dañarse o dañar otros circuitos.

Los tres estados de salida de estas compuertas son:

(a) Cero a la salida cuando el transistor inferior en el poste totem esta en saturación y el superior esta en corte.
(b) Uno a la salida cuando el transistor inferior en el poste totem esta en corte y el superior esta en saturación.
(c) Alta impedancia a la entrada que permite una conexión directa de muchas salidas en una línea común.

Investigar en los manuales de TTL el funcionamiento de los circuitos 74LS125 y 74LS126.

PRACTICA CINCO.Unidad Lógica-Aritmética (ALU)

Investigar el número del circuito integrado que contiene un ALU de cuatro bits. Conectarlo y realizar las operaciones lógicas que indica su tabla de verdad.

El ALU de cuatro bits puede realizar 16 operaciones lógicas y 16 operaciones aritméticas.

PRACTICA SEIS. Decodificador de BCD a siete segmentos y display

Investigar el funcionamiento de los ICs 74LS47 y 74LS48 que son decodificadores a 7 segmentos.

Conectar los ICs y probarlos. Para esto se tendrán que usar dos tipos diferentes de displays. Display de cátodo común, y display de ánodo común. Investiga su funcionamiento.

El decodificador BCD-a-siete-segmentos es un circuito combinacional que acepta un dígito decimal en BCD y genera las salidas apropiadas para la selección de segmentos en un display usado para mostrar el dígito decimal.

PRACTICA SIETE. Unidad Lógica-Aritmética con display de siete segmentos

Esta práctica consiste en integrar las prácticas cinco y seis en una sola. Las salidas del IC 74LS181 serán las entradas del decodificador de tal manera que los resultados de las operaciones realzadas con el ALU se muestren en el display.

PRACTICA OCHO. Sumador completo

Implementar el circuito que se muestra en la figura 7.3.4 y verificar la tabla de verdad del mismo.

PRACTICA NUEVE. Mapas de Karnaugh

Diseñar un restador con el método de Karnaugh e implementar el circuito de la misma forma que se hizo para el sumador. ¿Cuál es la diferencia entre un sumador y un restador?.

PRACTICA DIEZ. Aplicación de multiplexores

El objetivo de esta práctica es comprobar que el alumno entiende el funcionamiento de los circuitos combinacionales, por lo tanto se pide al alumno que diseñe un probador de circuitos integrados.

Un probador de circuitos integrados es un dispositivo que prueba si el CI funciona perfectamente, esto es que para todas y cada una de sus entradas posibles la salida sea correcta. Supóngase que se quiere probar el 7408, si las entradas de una de las compuertas es 00, la salida será 0, con un uno que haya a la entrada la salida será 1. El probador de ICs tiene que indicar si el IC esta dando la salida correcta.

Deberán utilizarse multiplexores y compuertas lógicas. El probador debe revisar los siguientes ICs: 74LS00, 74LS08, 74LS32, 74LS86.

El probador deberá tener cuatro entradas de selección, 2 para las entradas de las compuertas y dos para seleccionar el chip que será probado.

El resultado se indicará con un LED. Si el IC funciona correctamente el LED deberá estar prendido, de otra manera se deberá apagar.

Cerebros de silicio.
Circuitos digitales combinacionales
se terminó de imprimir en febrero de 1994
en los talleres de Editorial Conexión Gráfica,
Libertad 1471, S.J., 44100 Guadalajara, Jal., México.
Tels.: 625-6512, 626-3192, 625-5153 Fax: 626-3104
La edición consta de 1,000 ejemplares.
La tipografía utilizada es *Dutch* (Times Roman)
en 7,8,9,11 y 16 puntos.
La edición estuvo a cargo de *Alfabeto Editores*, tel. 625-58-25.
Ventas en el Departamento de Extensión Universitaria del ITESO.
Tels. 669-34-85 y 669-34-80 Fax 669-34-81
Guadalajara, Jal., México.

Este texto es la primera parte de un estudio teórico-práctico sobre los circuitos electrónicos digitales. Los temas expuestos a lo largo de este libro fueron elaborados para un curso dirigido a los alumnos de Ingeniería en Sistemas Computacionales del ITESO. A lo largo del curso se pretendió que, sin ningún prerrequisito, los alumnos finalizaran la experiencia académica sabiendo cómo se diseña un sistema dirigido y cómo funcionan los circuitos digitales, tanto en la teoría como en la práctica.

Patricia Calderón es Ingeniero en Comunicación y Electrónica por la Universidad de Guadalajara. Ha tomado los cursos: Microprocesadores, en Rochester, Mn.; Tópicos avanzados en lenguaje de programación orientado a objetos, en la Universidad de Loyola en Los Angeles, Ca.; Métodos operacionales, en la Universidad Autónoma Metropolitana, México, D.F. Ha sido profesora de álgebra, sistemas, geometría, circuitos digitales e ingeniería de programación en el ITESO y la Universidad de Guadalajara. Coordinadora de la carrera de Ingeniería en Sistemas Computacionales del ITESO, de 1991 a 1993, y actualmente profesora de ingenirería de programación II y circuitos digitales en la División de Ingeniería.

textos iteso